

# Per-se Privacy Preserving Distributed Optimization

P. C. Weeraddana, *Member, IEEE*, G. Athanasiou, *Member, IEEE*,  
M. Jakobsson, *Student Member, IEEE*, C. Fischione, *Member, IEEE*,  
and J. S. Baras, *Fellow, IEEE*

## Abstract

Distributed optimization is a fundamental mathematical theory for parallel and distributed systems. Several applications are normally designed based on such a theory, where parties cooperatively exchange messages with little or no central coordination to achieve some goals. In many situations, the transactions among the parties must be private, such as among members of social networks, hospitals, companies in a free market, banks, and state governments, to mention a few. Existing privacy preserving solution methods for optimization problems are mostly based on cryptographic procedures and thus have the drawback substantial computational complexity, which is infeasible for large scale networks. The availability of distributed optimization solution methods that are private *per se* are therefore highly desirable and sometimes the only ones viable. Surprisingly, little attention has been devoted thus far to the development of a general theory for such privacy preserving distributed optimization. In this survey paper, a new general framework of existing transformation based mechanisms for privacy preserving distributed optimization is presented. The privacy preserving properties that are inherent in the classical decomposition techniques, such as primal decomposition, dual decomposition and state-of-the-art methods, such as alternating direction method of multipliers are investigated. A number of examples is provided to illustrate the need of a new theory of per-se privacy preserving optimization. It is concluded that the theory is still in its infancy and that huge benefits can be achieved by a substantial development.

## Index Terms

Privacy, distributed optimization, parallel and distributed computation.

P. C. Weeraddana, G. Athanasiou, M. Jakobsson, and C. Fischione with Electrical Engineering, KTH Royal Institute of Technology, Stockholm, Sweden. J. C. Baras is with the Department of Electrical and Computer Engineering, University of Maryland, USA

## I. INTRODUCTION

Several real-world optimization problems involve parties or nodes interacting via some networks that must collaborate to solve an optimization problem for mutual benefit [1], [2]. For example, in the business sector, independent companies need to interact for completing a common business and thus have to work together to optimize their joint operations. Normally, optimization solvers require much public data sharing among the parties, which may substantially hinder or prevent the cooperation for optimization due to privacy concerns. The fundamental question is how to solve optimization problems among parties that would receive much benefit by collaboration and yet are unwilling to share their data without preserving privacy.

Cryptography is the standard approach to preserve privacy in distributed optimization solvers [3]. However, cryptography may introduce substantial overhead among the nodes due to the exchange of security information and coordination. Cryptography is prone to attack by third parties who may inadvertently own the cryptographic keys. Moreover, for large networks, namely for networks with a large number of parties, it might be prohibitive if not impossible to use cryptography. Therefore, it is highly desirable to have distributed solution algorithms without introducing extra coordination or overhead and that naturally preserve privacy in the transactions.

The availability of these distributed solution algorithms posses many appealing merits, which are desirable in practice, e.g., efficiency, scalability, natural (geographical) distribution of problem data. More importantly, they are per-se privacy preserving without requiring any extra coordination or overhead. In this paper, we propose a systematic study of methods ensuring both the efficient solution of distributed optimization problems and the privacy of the associate transactions. We further motivate the need of such a theory by some simple illustrative examples.

### A. Motivating examples

Here, we present some real-world examples that motivate and highlight the strong need of a per se privacy preserving optimization theory.

---

*Example 1 (Privacy preserving data mining [4]):* Suppose that there are two different government agencies, a local Agency 1 and a nation-wide Agency 2 that store their information at their databases. The general

goal is that both agencies have to collaborate and provide a joint classification of possible security threats. The main difficulty in this collaborative procedure is that several lower clearance level requirements at Agency 1 prevents Agency 2 from granting Agency 1 open access to the data base of Agency 2. Moreover, even if Agency 1 can gain temporary access to the database of Agency 2, the databases of these agencies are restricted to their current physical locations due to security policies. This is a typical example of distributed data mining, which relies entirely on distributed optimization. The example can be easily generalized to many agencies. This examples can be solved by using the per-se privacy preserving methods proposed in § V.

---

*Example 2 (Production line problem [5]):* We assume that Company 1 produces and sells some products (e.g., bags, etc) with various quality. These products can provide different profit levels to the company as well. The general problem is to determine how many pieces of each kind of product must the company produce in order to maximize the profit, assuming that all the made products will be bought. At some point, Company 1 would like to increase its profit and, therefore, outsources one or more production stages to another company, Company 2. However, Company 2 is not willing to disclose its capacity or its production times to Company 1. The only thing that Company 2 would like to share is the charging cost of each production stage. How can Company 1 decide if the deal is worthwhile based on this limited information? This is case of production line distributed optimization, which can be easily generalized to many companies. This examples can be solved in a per-se privacy preserving manner by using methods in § IV,V,III.

---

These real-world problems could be solved by applying cryptography, as proposed in literature. However, a per-se preserving optimization approach can be instead used with much more efficiency, as argued in this paper. In the following, we briefly present some representative approaches and highlight our original contribution.

## *B. Background and Contribution*

Existing privacy preserving methods used in distributed optimization solvers assume that nodes or parties of a network own some constraints and the overall network wants to optimize a cost function that may be given by the composition of functions belonging to the nodes. Note that in general, the network cost function may be given by a separable composition of the nodes's cost functions, or by a non-separable composition. The classic distributed optimization problem solver consists in running at the nodes algorithms that transmit to other nodes or network coordinator, data, such as decision variables, and receive by the network or by the coordinator back further information to iterate and reach eventually the computation of the optimal solution [1]. There are

essentially two approaches: cryptographic methods, and non cryptographic methods. The second approach, can then be further divided into algebraic transformation-based methods, which we survey and generalize in this paper, and potential decomposition based approaches, which we investigate substantially in this paper.

Cryptographic primitives include secure multiparty computations [6], pseudo random generator [7], [8], homomorphic encryption [9], [10], [11]. These methods use cryptographic tools such as zero-knowledge [12], oblivious transfer [13], 1-out-of-n oblivious transfer [14], oblivious evaluation of polynomials [15], secret sharing [16], and threshold cryptography [17]. In general, the area of cryptography-based privacy preserving optimization is very well investigated [3], [18]--[21]. In the context of optimization problems, cryptographic tools are used for securely perform iterations of well known *simplex* algorithm and *interior-point* algorithm so that the sensitive data is not disclosed during the methods, see [22]--[24] for secure simplex variants and [5] for secure variants of interior-point method. In terms of security, those cryptographic methods are highly desirable, though they are highly unfavorable in terms of computational complexity and efficiency [5].

Developing solution methods based on algebraic transformations that preserve privacy of data *without* cryptography has attracted the interest of the research community [4]--[6], [25]--[31]. We refer to those approaches as *transformation methods* in general. The key idea of these is to use algebraic manipulations to disguise the original problem into an equivalent problem so that the private data of each node is hidden. However, in these interesting papers, only some specific problems have been considered and there is not any attempt to establish a systematic approach. We strongly believe that the transformation based privacy preserving approaches still need to be systematically investigated. Moreover, decomposition methods [1], [32], and state-of-the-art distributed optimization methods, such as alternating direction method of multipliers (ADMM) [33] are less investigated in the context of privacy preserving problem solution methods. However, these approaches posses inherent privacy preserving properties, and therefore have a huge potential to handle privacy issues in a more efficient manner.

Given the drawbacks of cryptographic approaches, investigating efficient and scalable methods

that are per-se privacy preserving is of crucial importance from a theoretical, as well as from a practical perspective. Therefore, in this paper, we place a greater emphasis on transformation methods and optimization approaches, such as decomposition methods and ADMM methods that possess inherent privacy preserving characteristics, as opposed to treatments based on well investigated cryptographic primitives. Of course, there are many papers that describe in details cryptographic methods and are extraneous to the main focus of this paper (see, e.g., [3]). In particular, we summarize the *existing* privacy preserving solution methods based on transformation approaches by a more generic canonical form. Then, we discuss in detail a general decomposition structure together with classical decomposition techniques, including primal decomposition and dual decomposition, as well as state-of-the-art method ADMM. Their inherent privacy preserving properties are investigated. The importance of these per-se privacy preserving methods is highlighted through a number of examples and their performance is compared in terms of efficiency, scalability, complexity and many others. To the best of our knowledge, this is the first paper that provides a general classification and comparison of per-se privacy-preserving methods in *optimization*, and introduces the inherent privacy properties of decomposition methods.

The rest of the paper is organized as follows. In § II we present some basic definitions that are useful for describing the properties of privacy preserving optimization. A general formulation to model the existing privacy preserving techniques is presented in § III. Optimization approaches, such as decomposition, ADMM methods in the context of privacy preserving optimization are extensively discussed in § III and V. In § VI, we provide a summary, including comparisons of different methods for privacy preserving and future directions. Conclusions are given in § VII.

**Notations:** Boldface lower case and upper case letters represent vectors and matrices respectively and calligraphy letters represent sets. The set of real  $n$ -vectors is denoted  $\mathbb{R}^n$ , and the set of real  $m \times n$  matrices is denoted  $\mathbb{R}^{m \times n}$ . The identity matrix is denoted by  $\mathbf{I}$ . The superscript  $(\cdot)^T$  stands for transpose. Vectors and matrices are delimited with square brackets, with the components separated by space. The  $i$ th submatrix of a matrix is usually denoted by using a subscript. We use *parentheses* to construct column vectors from comma separated lists, e.g.,  $(\mathbf{a}, \mathbf{b}, \mathbf{c}) = [\mathbf{a}^T \ \mathbf{b}^T \ \mathbf{c}^T]^T$ . We denote by  $\|\mathbf{x}\|_1$  the  $\ell_1$ -norm and by  $\|\mathbf{x}\|_2$   $\ell_2$ -norm of the vector  $\mathbf{x}$ .

## II. PER-SE PRIVACY PRESERVING DEFINITIONS

Many standard privacy/security conventional definitions are already adopted in cryptographic literature; see for example [6], [34]--[38]. However, such definitions are not directly applied or adopted in the context of transformation based and decomposition based optimization methods. This is because the key mechanisms used for preserving the privacy in cryptographic protocols are entirely different from those that are to be used in optimization based approaches. Theoretical foundations for defining the privacy of optimization methods deserve attention in its own right. However, to highlight the appealing privacy preserving properties associated with optimization based approaches, and to provide a cohesive discussion, we give here some *basic* definitions.

*Definition 1 (Optimization problem):* We consider the following standard notation to describe the problem of finding a point  $\mathbf{x}$  that minimizes the function  $f_0(\mathbf{x})$  subject to a number of inequality and equality constraints:

$$\begin{aligned} & \text{minimize} && f_0(\mathbf{x}) \\ & \text{subject to} && f_i(\mathbf{x}) \leq 0, \quad i = 1, \dots, q \\ & && h_i(\mathbf{x}) = 0, \quad i = 1, \dots, p. \end{aligned} \tag{1}$$

We call (1), an optimization problem. Here  $f_0 : \mathbb{R}^n \rightarrow \mathbb{R}$  is called the *objective function*,  $f_i(\mathbf{x}) \leq 0$ ,  $i = 1, \dots, q$  are called *inequality constraints* with the associated inequality constraint functions  $f_i : \mathbb{R}^n \rightarrow \mathbb{R}$ ,  $i = 1, \dots, q$ ,  $h_i(\mathbf{x}) = 0$ ,  $i = 1, \dots, p$  are called *equality constraints* with the associated equality constraint functions  $h_i : \mathbb{R}^n \rightarrow \mathbb{R}$ ,  $i = 1, \dots, p$ , and  $\mathbf{x} = (x_1, \dots, x_n) \in \mathbb{R}^n$  is called the *optimization variable* [39]. The terms optimization variable, decision variable, and primal variable are all synonymous.

*Definition 2 (convex problem):* A *convex* optimization problem is a one of the same form as (1) provided some assumptions hold. In particular, the convex problem is

$$\begin{aligned} & \text{minimize} && f_0(\mathbf{x}) \\ & \text{subject to} && f_i(\mathbf{x}) \leq 0, \quad i = 1, \dots, q \\ & && \mathbf{C}\mathbf{x} - \mathbf{d} = \mathbf{0}, \end{aligned} \tag{2}$$

where the functions  $f_i$ ,  $i = 0, \dots, q$  are *convex* and  $h_i$ ,  $i = 1, \dots, p$  are *affine*, i.e., the equality constraint functions are given by  $\mathbf{C} \in \mathbb{R}^{p \times n}$  with  $\text{rank}(\mathbf{C}) = p$  and  $\mathbf{d} \in \mathbb{R}^p$ . The optimization variable is  $\mathbf{x} = (x_1, \dots, x_n) \in \mathbb{R}^n$ . Typically, we have  $p < n$  in practice, otherwise the only *potential* solution is  $\mathbf{C}^\dagger \mathbf{d}$ , where  $\mathbf{C}^\dagger$  is called the pseudo-inverse of  $\mathbf{C}$  (see [39, § A.5.4]).

*Definition 3 (K-party environment):* A set of  $K$  parties is called an  $N$ -party environment.

Through out this paper, we consider problems of the form (1)–(2), which are to be solved in an  $K$ -party environment via coordination among the parties. The global variable  $\mathbf{x}$ , objective function  $f_0$ , inequality functions  $f_i$   $i = 1, \dots, q$ , equality functions  $h_i$   $i = 1, \dots, m$ , different subset of components of  $\mathbf{x}$ , different subset of functions, and/or function definitions themselves can be spread out among  $K$  parties. We usually use the terms *ownership*, *private data* to indicate clearly spreading among  $K$  parties such functions, data, etc. Note that the terms *party* and *entity* are often used interchangeably in this paper.

---

*Example 3:* Consider the case  $f_0(\mathbf{x}) = \sum_{i=1}^K \mathbf{c}_i^T \mathbf{x}_i$ , where  $\mathbf{c}_i \in \mathbb{R}^{n_i}$ ,  $\mathbf{x}_i \in \mathbb{R}^{n_i}$  with  $\sum_{i=1}^K n_i = n$ , and  $\mathbf{x} = (\mathbf{x}_1, \dots, \mathbf{x}_K)$ . Here, the objective function  $f_0$  and the optimization variable  $\mathbf{x}$  is spread out among  $K$ -parties, such that  $\mathbf{c}_i$  and  $\mathbf{x}_i$  are owned by party  $i$ .

---

*Definition 4 (Semi-honest adversary):* In a multi-party environment, an entity involved in solving a global optimization problem of the form (1), centrally or in a coordinated fashion with other entities, is called a semi-honest adversary, if it executes its prescribed computations correctly, but keeps a record of all information it receives from other parties and tries to learn properties of others' private data.

The definition above is inspired from [40] and is similar to the *good* and the *passive* adversary models given in [41] and [42], respectively. In a multi-party environment, there can be more than one adversaries. We consider the *semi-honest adversary* model throughout this paper.

*Definition 5 (Inputs and outputs of a convex problem):* Consider the convex problem (2). We call the *set of problem parameters*, the terms  $\mathbf{C}, \mathbf{d}$ , those ones required to define the functions  $f_0, f_i$  and/or  $\mathbf{C}, \mathbf{d}$  together with the functions themselves as *inputs* of problem (2). Moreover, we call the solution and the optimal value of problem (2) as *outputs*.

---

*Example 4:* Consider the following linear program:

$$\begin{aligned} & \text{minimize} && \mathbf{c}^T \mathbf{x} \\ & \text{subject to} && \mathbf{A} \mathbf{x} \leq \mathbf{b} \\ & && \mathbf{x} \geq \mathbf{0} , \end{aligned} \tag{3}$$

where the variable is  $\mathbf{x} \in \mathbb{R}^n$  and problem data are  $\mathbf{c} \in \mathbb{R}^n$ ,  $\mathbf{A} \in \mathbb{R}^{m \times n}$ , and  $\mathbf{b} \in \mathbb{R}^m$ . Suppose  $\mathbf{x}^*$  solves the problem. The parameters  $\{\mathbf{c}, \mathbf{A}, \mathbf{b}\}$  are the inputs of the problem and  $\{\mathbf{x}^*, p^*\}$  are the outputs of the problem,

where  $p^* = \mathbf{c}^T \mathbf{x}^*$ .

---

*Definition 6 (Input privacy):* We say that the mechanism for solving an optimization problem yields input privacy for a subset  $\mathcal{C}$  of the input if during the solution method stages, *only* a transformed variant of the original objects in  $\mathcal{C}$  is available to the adversary and such that recovering the original  $\mathcal{C}$  is impossible without the knowledge of the transformation. Moreover, we say that the solution method for an optimization problem yields input privacy for  $\mathcal{C}$  also if  $\mathcal{C}$  is *unknown* to the adversary.

---

*Example 5:* Consider once again the linear program (3), which is to be solved by Alice and Bob. The input ownership is as follows: Alice owns  $\mathbf{c}$  and Bob owns  $\{\mathbf{A}, \mathbf{b}\}$ . Now suppose they use the following solution procedure: instead of the input  $\mathbf{c}$ , Alice uses  $\hat{\mathbf{c}}$ , where  $\hat{\mathbf{c}} = \alpha \mathbf{c}$  and  $\alpha$  is a positive scalar known by Alice only. Similarly, instead of the input  $\{\mathbf{A}, \mathbf{b}\}$ , Bob uses  $\{\hat{\mathbf{A}}, \hat{\mathbf{b}}\}$ , where  $\hat{\mathbf{A}} = \beta \mathbf{A}$ ,  $\hat{\mathbf{b}} = \beta \mathbf{b}$ , and  $\beta$  is a positive scalar known by Bob only. The result is the following optimization problem:

$$\begin{aligned} & \text{minimize} && \hat{\mathbf{c}}^T \mathbf{x} \\ & \text{subject to} && \hat{\mathbf{A}} \mathbf{x} \leq \hat{\mathbf{b}} \\ & && \mathbf{x} \geq \mathbf{0} , \end{aligned} \tag{4}$$

where the variable is  $\mathbf{x}$ . Problem (3) and (4) are clearly equivalent, because  $\hat{\mathbf{A}} \mathbf{x} \leq \hat{\mathbf{b}} \Leftrightarrow \mathbf{A} \mathbf{x} \leq \mathbf{b}$  and minimizing  $\hat{\mathbf{c}}^T \mathbf{x}$  is identical to minimizing  $\mathbf{c}^T \mathbf{x}$  with  $\alpha > 0$ . Either Alice or Bob can use any linear programming solver for solving the problem. This solution procedure yields input privacy for Alice's data  $\mathbf{c}$  and for Bob's data  $\{\mathbf{A}, \mathbf{b}\}$  because without knowing  $\beta$ , Alice cannot recover Bob's data  $\{\mathbf{A}, \mathbf{b}\}$ , by using  $\{\hat{\mathbf{A}}, \hat{\mathbf{b}}\}$ , and in addition, without knowing  $\alpha$ , Bob cannot recover Alice's data  $\mathbf{c}$ , by using  $\hat{\mathbf{c}}$ .

---

*Definition 7 (Output Privacy):* We say that the mechanism for solving an optimization problem yields output privacy for a subset  $\mathcal{X}$  of the output if at the end, *only* a transformed variant of  $\mathcal{X}$  is available to the adversary such that recovering the original  $\mathcal{X}$  is impossible without knowing the transformation. Moreover, we say that the solution method for an optimization problem yields output privacy also if  $\mathcal{X}$  is *unknown* to the adversary at the end.

More examples will be discussed in § III and § IV. We emphasize that investigating a comprehensive treatment of a set of mathematical definitions is out of the scope of this paper. We believe that the basic definitions (4)-(7) given above are sufficient for surveying the appealing aspects and properties of privacy preserving optimization approaches presented in this paper.



### III. TRANSFORMATION BASED METHODS FOR PRIVACY PRESERVING

In this section, existing transformation based methods proposed for privacy preserving linear programming [5, § 6-7] [26], [27], [43] [30, § 4] [25], [28], [31], [44], as well as nonlinear programming [45], [46] are considered. The number of parties involved in a problem can be either two [5], [25], [30], [31], [43], [44] or more [26]--[28], [45], [46]. In the sequel, we provide a generic description for such privacy preserving solution methods proposed in the literature.

Transformation method is directly based on the notion of equivalence of optimization problems [39, § 4.1.3]. Two problems are called to be *equivalent* if, from a solution of one, a solution of the other is readily derived. There are many general transformations that yield equivalent problems. We next propose *two* transformations, which captures all the problem formulations in [5], [25]--[28], [30], [31], [43]--[46].

#### A. Transformation via Change of Variable

Suppose the original problem to be solved in a privacy preserving manner is given by (2). We denote by  $\mathcal{D}$  the set of points for which the objective and all constraint functions are defined, or domain of problem (2), i.e.,  $\mathcal{D} = \bigcap_{i=0}^q \text{dom } f_i \cap \mathbb{R}^{n-1}$ . Let  $\phi : \mathbb{R}^m \rightarrow \mathbb{R}^n$  be a function, with image *covering* the problem domain  $\mathcal{D}$ . Now we perform the following change of variables:

$$\mathbf{x} = \phi(\mathbf{z}) . \quad (5)$$

The resulting problem is given by

$$\begin{aligned} & \text{minimize} && f_0(\phi(\mathbf{z})) \\ & \text{subject to} && f_i(\phi(\mathbf{z})) \leq 0, \quad i = 1, \dots, q \\ & && \mathbf{C}\phi(\mathbf{z}) - \mathbf{d} = \mathbf{0} , \end{aligned} \quad (6)$$

where the variables are  $\mathbf{z} \in \mathbb{R}^m$ . It follows that if  $\mathbf{x}$  solves problem (2), then  $\mathbf{z} = \phi^{-1}(\mathbf{x})$  solves problem (6). Moreover, if  $\mathbf{z}$  solves problem (6), then  $\mathbf{x} = \phi(\mathbf{z})$  solves problem (2). Thus, the two problems are equivalent. Such a transformation is used to ensure privacy, as we see next.

#### Privacy Properties

If the function  $\phi$  is chosen appropriately, then any solution method applied to the transformed problem (via change of variables) can yield the *input privacy* for many inputs (except for  $\mathbf{d}$ ) via

<sup>1</sup>For many considered problem formulations in this paper, the domain  $\mathcal{D} = \mathbb{R}^n$ .

the function compositions

$$\begin{aligned}\hat{f}_i(\mathbf{z}) &= f_i(\phi(\mathbf{z})) , \quad \text{dom } \hat{f}_i = \{\mathbf{z} \in \text{dom } \phi \mid \phi(\mathbf{z}) \in \text{dom } f_i\} , \quad i = 0, \dots, q , \\ \hat{h}_i(\mathbf{z}) &= \mathbf{C}\phi(\mathbf{z}) - \mathbf{d} , \quad \text{dom } \hat{h}_i = \{\mathbf{z} \in \text{dom } \phi \mid \phi(\mathbf{z}) \in \mathbb{R}^n\} .\end{aligned}$$

The output privacy for the optimal solution is attained by the definition of  $\phi$  (see (5)). In the sequel, we highlight these privacy properties by using some examples. first, lets see some choices for the function  $\phi$ .

---

*Example 6:*

- **Scaling:** Here, we simply use the change of variable  $\mathbf{x} = \phi(\mathbf{z}) = a\mathbf{z}$ , where  $a$  is a scalar.
- **Translation:** Here, we use the following change of variable,  $\mathbf{x} = \phi(\mathbf{z}) = \mathbf{z} + \mathbf{a}$ , where  $\mathbf{a} \in \mathbb{R}^n$ .
- **Affine transformation:** This is a generalization of both scaling and translation. Specifically, we use the following change of variable,  $\mathbf{x} = \phi(\mathbf{z}) = \mathbf{B}\mathbf{z} + \mathbf{a}$ , where  $\mathbf{B} \in \mathbb{R}^{n \times m}$  is a *full* rank matrix with  $\text{rank}(\mathbf{B}) = n$  and  $\mathbf{a} \in \mathbb{R}^n$ . Thus, a particular inverse transformation  $\phi^{-1} : \mathbb{R}^n \rightarrow \mathbb{R}^m$  is given by

$$\mathbf{z} = \phi^{-1}(\mathbf{x}) = \mathbf{B}^\dagger \mathbf{x} - \mathbf{B}^\dagger \mathbf{a} ,$$

where  $\mathbf{B}^\dagger = \mathbf{B}^T(\mathbf{B}\mathbf{B}^T)^{-1}$ , which is typically known as pseudo-inverse or Moore-Penrose inverse.

- **Nonlinear transformations:** One example is as follows,  $x_i = \phi_i(z_i) = a_i \exp(z_i)$ , where  $a_i > 0$ .
- 

We see that all the approaches [5], [25]--[27], [30], [31], [43]--[46] have used *change of variables* (affine transformations) as one of the mechanism for privacy preserving in their proposed solution methods. To illustrate this, we consider few key examples from the literature.

---

*Example 7 (Computation outsourcing in the cloud, a 2-party environment [44]):* Suppose a cloud customer wants to outsource to the cloud his linear program

$$\begin{aligned}& \text{minimize} && \mathbf{c}^T \mathbf{x} \\ & \text{subject to} && \mathbf{A}\mathbf{x} = \mathbf{b} \\ & && \mathbf{B}\mathbf{x} \geq \mathbf{0} ,\end{aligned}\tag{7}$$

where the variable is  $\mathbf{x} \in \mathbb{R}^n$  and the problem data are  $\mathbf{c} \in \mathbb{R}^n$ ,  $\mathbf{A} \in \mathbb{R}^{m \times n}$  with  $m < n$ , nonsingular  $\mathbf{B} \in \mathbb{R}^{n \times n}$ , and  $\mathbf{b} \in \mathbb{R}^m$ . The customer does not want to reveal problem data  $\mathbf{c}, \mathbf{A}, \mathbf{b}, \mathbf{B}$  and the solution  $\mathbf{x}^*$  of the problem to the cloud, i.e., input privacy for  $\{\mathbf{c}, \mathbf{A}, \mathbf{b}, \mathbf{B}\}$  and output privacy for  $\mathbf{x}^*$  is the requirement.

The cloud customer then uses the affine transformation [44, § III-C]

$$\mathbf{x} = \phi(\mathbf{z}) = \mathbf{M}\mathbf{z} - \mathbf{r} ,$$

where  $\mathbf{M} \in \mathbb{R}^{n \times n}$  is a nonsingular matrix and  $\mathbf{r} \in \mathbb{R}^n$  is a vector, both known by the customer only. The

equivalent problem outsourced by the customer to the cloud is given by

$$\begin{aligned} & \text{minimize} \quad \hat{\mathbf{c}}^T \mathbf{z} \\ & \text{subject to} \quad \hat{\mathbf{A}} \mathbf{z} = \hat{\mathbf{b}} \\ & \quad \quad \quad \hat{\mathbf{B}} \mathbf{z} \geq \mathbf{0} , \end{aligned} \tag{8}$$

where the variable is  $\mathbf{z} \in \mathbb{R}^m$  and the problem input is  $\hat{\mathbf{c}} = \mathbf{M}^T \mathbf{c}$ ,  $\hat{\mathbf{A}} = \mathbf{M} \mathbf{A}$ ,  $\hat{\mathbf{b}} = \mathbf{b} + \mathbf{A} \mathbf{r}$ , and  $\hat{\mathbf{B}} = \mathbf{M} \mathbf{B}$ . The cloud computes the optimal solution of problem (8), which we denote by  $\mathbf{z}^*$ .

The sensitive inputs of problem (7),  $\{\mathbf{c}, \mathbf{A}, \mathbf{b}, \mathbf{B}\}$ , cannot be recovered by a potential adversary or the cloud because the matrix  $\mathbf{M}$  and the column vector  $\mathbf{r}$  are not known to the cloud. For the same reasons, the cloud cannot construct the sensitive output  $\mathbf{x}^*$  by using  $\mathbf{z}^*$ . Thus, the solution procedure yields both input privacy and output privacy.

The problem data (input) structure can be exploited so that change of variables can be applied to develop privacy preserving solution methods in multi-party environments [26], [27], [45], [46].

*Example 8 (Classification, a multi-party situation [46]):* Consider the following problem:

$$\begin{aligned} & \text{minimize} \quad \|\mathbf{v}\|_1 + \|\bar{\mathbf{B}} \mathbf{x}\|_1 \\ & \text{subject to} \quad \mathbf{D}(\mathbf{A} \mathbf{x} - \mathbf{1} u) + \mathbf{v} \geq \mathbf{1} \\ & \quad \quad \quad \mathbf{v} \geq \mathbf{0} , \end{aligned} \tag{9}$$

where the variables are  $\mathbf{x} \in \mathbb{R}^n$ ,  $\mathbf{v} \in \mathbb{R}^m$ , and  $u \in \mathbb{R}$ . Here the problem parameters  $\mathbf{D} = \text{diag}(d_1, \dots, d_m)$  with  $d_i \in \{-1, +1\}$  is known as *class matrix*,  $\mathbf{A} \in \mathbb{R}^{m \times n}$  is the *feature matrix* [45], and  $\bar{\mathbf{B}} \in \mathbb{R}^{\bar{m} \times n}$ . The goal is to find the *public linear classifier* given by  $(\mathbf{x}^*)^T \mathbf{y} - u^* = 0$ , where  $(\mathbf{x}^*, \mathbf{v}^*, u^*)$  is the solution of problem (9). Suppose  $\mathbf{A}$  and  $\mathbf{x}$  are partitioned as follows:

$$\mathbf{A} = [\mathbf{A}_1, \dots, \mathbf{A}_q] , \quad \mathbf{x} = (\mathbf{x}_1, \dots, \mathbf{x}_q) ,$$

where  $\mathbf{A}_i \in \mathbb{R}^{m \times n_i}$ ,  $\mathbf{x}_i \in \mathbb{R}^{n_i}$ , and  $\sum_{i=1}^q n_i = n$ . Here we have  $q$  entities and each submatrix  $\mathbf{A}_i$  is owned by  $i$ th entity. The  $i$ th entity does not want to reveal its problem data  $\mathbf{A}_i$  and the solution  $\mathbf{x}_i^*$  to the other entities  $j$  ( $j \neq i$ ), who might act as the adversaries. Thus, input privacy for  $\{\mathbf{A}_i\}_{i=1, \dots, q}$  and output privacy for  $\{\mathbf{x}_i^*\}_{i=1, \dots, q}$  is the requirement.

All the entities then use the transformation [45, § 2]  $\mathbf{x} = \phi(\mathbf{z}) = \mathbf{B}^T \mathbf{z}$ , where  $\mathbf{B} \in \mathbb{R}^{\bar{m} \times n}$  is a full rank matrix with  $\text{rank}(\mathbf{B}) = n$  and is horizontally partitioned as follows:  $\mathbf{B} = [\mathbf{B}_1, \dots, \mathbf{B}_q]$ . Here the submatrix  $\mathbf{B}_i \in \mathbb{R}^{\bar{m} \times n_i}$  is known only by the  $i$ th entity. Now consider the closely related problem (see [45, equ. 2.6])

$$\begin{aligned} & \text{minimize} \quad \|\mathbf{v}\|_1 + \|\mathbf{z}\|_1 \\ & \text{subject to} \quad \mathbf{D} \left( \left( \sum_{i=1}^q \mathbf{A}_i \mathbf{B}_i^T \right) \mathbf{z} - \mathbf{1} u \right) + \mathbf{v} \geq \mathbf{1} \\ & \quad \quad \quad \mathbf{v} \geq \mathbf{0} , \end{aligned} \tag{10}$$

with the variables are  $\mathbf{z} \in \mathbb{R}^m$ ,  $\mathbf{v} \in \mathbb{R}^m$ , and  $u \in \mathbb{R}$ . Let us denote the solution by  $(\mathbf{z}^*, \mathbf{v}^*, u^*)$ . Provided  $\bar{\mathbf{B}} = \mathbf{B}(\mathbf{B}^T \mathbf{B})^{-1}$ , problem (9) is equivalent to problem (10). Note that in the above solution method no entity  $i$  reveals its sensitive input  $\mathbf{A}_i$  and its private submatrix  $\mathbf{B}_i$  to others. Instead, entity  $i$  needs to make public

only the product  $\mathbf{A}_i \mathbf{B}_i^T$ . As a result, no entity  $j$  ( $\neq i$ ) can recover  $\mathbf{A}_i$  and  $\mathbf{x}_i^* = \mathbf{B}_i^T \mathbf{z}^*$  without knowing  $\mathbf{B}_i$ , i.e., the input privacy and the output privacy is obtained. Moreover, for any new  $\mathbf{y} = (\mathbf{y}_1, \dots, \mathbf{y}_q) \in \mathbb{R}^n$ , each entity  $i$  makes public  $\mathbf{B}_i \mathbf{y}_i$  from which the public linear classifier is computed as follows:

$$(\mathbf{z}^*)^T \mathbf{B} \mathbf{y} - u^* = (\mathbf{z}^*)^T \sum_{i=1}^q \mathbf{B}_i \mathbf{y}_i - u^* = 0 .$$


---

As listed in Example 6, it is also possible to use nonlinear change of variables for developing privacy preserving solution methods. To see this let us consider the following example:

---

*Example 9 (Nonlinear transformation, a 2-party situation):* Alice wants to outsource to an untrusted party the problem

$$\begin{aligned} & \text{minimize} && \sum_{i=1}^n (\alpha_i / x_i) \\ & \text{subject to} && \sum_{i=1}^n \beta_i x_i^2 \leq \gamma \\ & && \mathbf{x} \geq \mathbf{0} , \end{aligned} \tag{11}$$

where the variable is  $\mathbf{x}$ . Here, the problem data are  $\alpha_i > 0$ ,  $\beta_i > 0$ , and  $\gamma > 0$ . Suppose Alice wants input privacy for problem data  $\{\alpha_i, \beta_i\}_{i=1, \dots, n}$ ,  $\gamma$ . By using the nonlinear change of variable given in Example 6, we have  $x_i = \phi_i(z_i) = \alpha_i \exp(z_i)$ . Next Alice can obtain the equivalent problem:

$$\begin{aligned} & \text{minimize} && \sum_{i=1}^n \exp(-z_i) \\ & \text{subject to} && \sum_{i=1}^n \lambda_i \exp(2z_i) \leq 1 , \end{aligned} \tag{12}$$

where the variable is  $\mathbf{z} = (z_1, \dots, z_n)$  and problem parameter  $\lambda_i = (\beta_i \alpha_i^2 / \gamma)$ . Now Alice can outsource problem (12) to the untrusted party. We can see that this solution procedure clearly yields input privacy for the sensitive input  $\{\{\alpha_i, \beta_i\}_{i=1, \dots, n}, \gamma\}$ . The solution  $\mathbf{x}^*$  of original problem is simply obtained by  $x_i^* = \alpha_i \exp(z_i^*)$ , where  $z_i^*$  is the solution of problem (12).

---

## B. Transformation of Objective and Constraint Functions

Let the original problem to be solved in a privacy preserving manner be given by (2). Suppose  $\psi_0 : \mathbb{D}_0 \subseteq \mathbb{R} \rightarrow \mathbb{R}$  is monotonically increasing, with domain covering the image of  $f_0$ , i.e.,  $\mathbb{D}_0 \supseteq \text{image } f_0$ . Moreover, suppose that for  $i = 1, \dots, q$ ,  $\psi_i : \mathbb{D}_i \subseteq \mathbb{R} \rightarrow \mathbb{R}$ , with  $\mathbb{D}_i \supseteq \text{image } f_i$ ,  $\psi_i(z) \leq 0$  if and only if  $z \leq 0$  and  $\psi : \mathbb{R}^p \rightarrow \mathbb{R}^m$  satisfies  $\psi(\mathbf{z}) = \mathbf{0}$  if and only if  $\mathbf{z} = \mathbf{0}$ .

The equivalent problem is clearly given by

$$\begin{aligned} & \text{minimize} && \psi_0(f_0(\mathbf{x})) \\ & \text{subject to} && \psi_i(f_i(\mathbf{x})) \leq 0, \quad i = 1, \dots, q \\ & && \psi(\mathbf{C}\mathbf{x} - \mathbf{d}) = \mathbf{0} , \end{aligned} \tag{13}$$

where the variable is  $\mathbf{x} \in \mathbb{R}^n$ . The optimal solution of problem (13) is identical to that of problem (2). The optimal value of problem (2),  $p^*$ , and that of problem (13),  $q^*$ , are related by

$$\psi_0(p^*) = q^* . \quad (14)$$

### Privacy Properties

With carefully chosen functions  $\psi_i$ , and  $\psi$ , any solution method for solving the transformed problem would yield *input privacy* for the input of problem (2) via the function compositions

$$\begin{aligned} \bar{f}_i(\mathbf{x}) &= \psi_i(f_i(\mathbf{x})) , \quad \text{dom } \bar{f}_i = \{\mathbf{x} \in \text{dom } f_i \mid f_i(\mathbf{x}) \in \text{dom } \psi_i\} , \quad i = 0, \dots, q , \\ \bar{h}_i(\mathbf{x}) &= \psi(\mathbf{C}\mathbf{x} - \mathbf{d}) , \quad \text{dom } \bar{h}_i = \mathbb{R}^n . \end{aligned}$$

Note that the optimal value  $p^*$  is not available to a potential adversary, though the optimal solution is. As a result, output privacy is attained for the optimal value, see (14). In the sequel, we give some examples to highlight these ideas.

*Example 10 (Scaling):* This idea is already presented in Example 5. Scaling is used in part to develop privacy preserving solution methods in references such as [5], [25], [30], [31], [43], [44]. Generally speaking, here all the functions  $\psi_i$ ,  $i = 0, \dots, q$  and  $\psi$  are linear (see (13)), i.e.,

$$\psi_i(z) = a_i z , \quad i = 0, \dots, q \quad \text{and} \quad \psi(\mathbf{z}) = \mathbf{B}\mathbf{z} , \quad (15)$$

where  $\{a_i\}_{i=0, \dots, q}$  are positive scalars and  $\mathbf{B} \in \mathbb{R}^{p \times p}$  is a diagonal matrix with nonzero diagonal entries, which are unknown to any potential adversary. Specifically, the generalized permutation matrix used in [31, § 4.1], [5, p. 69], the scalar  $\gamma$  used in [44, III-B-3], and the positive diagonal matrix  $S$  used in [43, § III] are identical to some scaling of the form (15).

*Example 11 (Least-squares problem, a multi-party situation):* Let us consider the basic but important problem

$$\text{minimize} \quad \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2 , \quad (16)$$

where the variable is  $\mathbf{x} \in \mathbb{R}^n$  and the problem data  $\mathbf{A} \in \mathbb{R}^{m \times n}$  with  $\text{rank}(\mathbf{A}) = n$  and  $\mathbf{b} \in \mathbb{R}^m$ .

Suppose  $\mathbf{A}$  and  $\mathbf{b}$  are partitioned as follows:

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_1^T & \dots & \mathbf{A}_q^T \end{bmatrix}^T , \quad \mathbf{b} = (\mathbf{b}_1, \dots, \mathbf{b}_q) ,$$

where  $\mathbf{A}_i \in \mathbb{R}^{m_i \times n}$ ,  $\mathbf{b}_i \in \mathbb{R}^{m_i}$ ,  $\sum_{i=1}^q m_i = m$ , and  $m_i > n$  for all  $i = 1, \dots, q$ . The submatrices  $\mathbf{A}_i$ ,  $\mathbf{b}_i$  are owned by  $i$ th entity and altogether  $q$  entities, who want the cloud to solve problem (16) for them. The  $i$ th entity does not want to reveal its problem data  $\mathbf{A}_i$  and  $\mathbf{b}_i$  to the other entities  $j$  ( $j \neq i$ ), and the cloud who might act as the adversaries. Thus, input privacy for  $\{\mathbf{A}_i, \mathbf{b}_i\}_{i=1, \dots, q}$  is the requirement.

The idea is to simply use this objective transformation  $\psi_0(z) = z^2$ , to yield

$$\text{minimize}_{\mathbf{x}} \|\mathbf{Ax} - \mathbf{b}\|_2^2 = \mathbf{x}^T \mathbf{A}^T \mathbf{A} \mathbf{x} - 2\mathbf{b}^T \mathbf{A} \mathbf{x} + \mathbf{b}^T \mathbf{b}$$

and to let the cloud solve the resulting equivalent problem above. Clearly, we have  $\mathbf{A}^T \mathbf{A} = \sum_{i=1}^q \mathbf{A}_i^T \mathbf{A}_i$ ,  $\mathbf{b}^T \mathbf{A} = \sum_{i=1}^q \mathbf{b}_i^T \mathbf{A}_i$ , and  $\mathbf{b}^T \mathbf{b} = \sum_{i=1}^q \mathbf{b}_i^T \mathbf{b}_i$ . Thus, the problem can be outsourced by simply each entity  $i$  making public only the products of matrices  $\{\mathbf{A}_i^T \mathbf{A}_i, \mathbf{b}_i^T \mathbf{A}_i, \mathbf{b}_i^T \mathbf{b}_i\}$ . Note that nobody can construct the sensitive inputs of entity  $i$   $\{\mathbf{A}_i, \mathbf{b}_i\}$  by knowing the public matrix products mentioned above. This means that the solution method for solving problem (16) yields input privacy for  $\{\mathbf{A}_i, \mathbf{b}_i\}_{i=1, \dots, q}$ .

---

*Example 12 (Horizontally Partitioned Linear Programs, a multi-party situation [28]):* The method proposed by the authors in [28] is built on the following equality constraint transformation:

$$\psi(\mathbf{z}) = \mathbf{Bz}, \quad (17)$$

where the  $\mathbf{B} \in \mathbb{R}^{m \times p}$  with  $m \geq p$  and  $\text{rank}(\mathbf{B}) = p$ . Thus, the following equivalence holds:

$$\mathbf{Ax} - \mathbf{b} = \mathbf{0} \Leftrightarrow \psi(\mathbf{Ax} - \mathbf{b}) \Leftrightarrow \mathbf{B}(\mathbf{Ax} - \mathbf{b}) = \mathbf{0}, \quad (18)$$

where  $\mathbf{A} \in \mathbb{R}^{p \times n}$  and  $\mathbf{b} \in \mathbb{R}^p$ . We can easily show (18) by using the pseudo-inverse of  $\mathbf{B}$ , which is given by  $\mathbf{B}^\dagger = (\mathbf{B}^T \mathbf{B})^{-1} \mathbf{B}^T$ . In reference [28], the authors exploit the partitioning of matrices  $\mathbf{A}, \mathbf{b}$  together with carefully chosen partitioned matrix  $\mathbf{B}$  to develop a solution method that yields input privacy for sensitive input (see [28, § 2] for details).

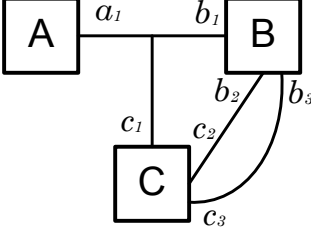
---

One can readily apply hybrid variants of the *transformation via change of variables* (see § III-A) and *transformation of objective and constraints* (see § III-B). Such hybrid variants can be found in the references [25], [30], [43], [44]. For example (roughly speaking), in [43, § III], the authors have first used the change of variable  $\mathbf{x} = \mathbf{Qz} - \mathbf{Qr}$ , where  $\mathbf{Q}$  is a generalized permutation matrix [31, § 4.1]. Then, for some constraints in the resulting equivalent problem, they have applied scaling by using  $\mathbf{Q}^{-1}$  and a positive diagonal matrix  $\mathbf{S}$ .

#### IV. PRIVACY PRESERVING IN DECOMPOSITION METHODS

In this section, we highlight important aspects of decomposition methods for designing privacy preserving solution algorithms. Although these techniques have been heavily used in the context of parallel and distributed optimization (see [1], [32], [33], [47], [48] and the references therein), we remark that they have not been investigated in the context of privacy preserving optimization. There are appealing *intrinsic* privacy preserving properties of these methods, as opposed to extrinsically acquired privacy in transformation based methods (see § III) and cryptographic

TABLE I  
DECOMPOSITION STRUCTURE: 3 ENTITIES, A, B, AND C WITH 3 NETS

	$\mathbf{y}_1 = a_1, \quad \mathbf{y}_2 = (b_1, b_2, b_3), \quad \mathbf{y}_3 = (c_1, c_2, c_3)$ $p = 7, \quad \mathbf{y} = (\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3) = (a_1, b_1, b_2, b_3, c_1, c_2, c_3) \in \mathbb{R}^7$ coupling: net 1 ( $z_1$ ): $a_1 = b_1 = c_1$ , net 2 ( $z_2$ ): $b_2 = c_2$ , net 3 ( $z_3$ ): $b_3 = c_3$ 3 nets $\Rightarrow \mathbf{z} = (z_1, z_2, z_3) \in \mathbb{R}^3$  $\mathbf{y} = \mathbf{E}\mathbf{z} = [\mathbf{E}_1^T \quad \mathbf{E}_2^T \quad \mathbf{E}_3^T]^T \mathbf{z}, \quad \text{where}$  $\mathbf{E}_1 = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}, \mathbf{E}_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \mathbf{E}_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
---	--

methods. In the sequel, we present concisely the theory associated with the classical techniques of primal and dual decomposition [1], [32]. Moreover, we highlight their use in privacy preserving optimization by a number of examples. Note that the solution methods based on decomposition relies heavily on the problem structure. Key techniques such as introduction of new variables, duality (see [39, § 4]) are often useful in such situations to facilitate decomposition as we will see in the sequel.

#### A. Primal Decomposition

Suppose there are  $K$  entities and the entity  $i$  has *private* variables  $\mathbf{x}_i \in \mathbb{R}^{n_i}$ , *shared* or *public* variables  $\mathbf{y}_i \in \mathbb{R}^{p_i}$  private objective function  $f_i : \mathbb{R}^{n_i} \rightarrow \mathbb{R}^{p_i}$ , and *private* constraint set  $\mathcal{C}_i \subseteq \mathbb{R}^{n_i} \times \mathbb{R}^{p_i}$ . The entities are coupled through public variables that require various scalar components of those to be equal. Such coupling or complications can be graphically shown by adding a hyper edge (or a net) that connects the entities for any scalar equality constraint among them. Let  $N$  be the number of nets in the system and  $\mathbf{z} = (z_1, \dots, z_N) \in \mathbb{R}^N$  be the common values of the public variables on the nets. For instance, the decomposition structure shown in Table I has 3 nets and  $\mathbf{z} = (z_1, z_2, z_3)$ . We denote by  $\mathbf{y} \in \mathbb{R}^p$  the concatenated public vectors, where  $\mathbf{y} = (\mathbf{y}_1, \dots, \mathbf{y}_K)$  and  $p = \sum_{i=1}^K p_i$  is the total number of *scalar* public variables (see Table I). By this notation, we let  $\mathbf{y} = \mathbf{E}\mathbf{z}$ , where

$$[\mathbf{E}]_{i,j} = E_{ij} = \begin{cases} 1 & [\mathbf{y}]_i \text{ is in net } j \\ 0 & \text{otherwise} . \end{cases} \quad (19)$$

We introduce a partition of  $\mathbf{E}$  as  $[\mathbf{E}_1^T \ \cdots \ \mathbf{E}_K^T]^T$ , where  $\mathbf{E}_i \in \mathbb{R}^{p_i \times N}$  denotes the block of  $\mathbf{E}$  associated with the entity  $i$ , so that  $\mathbf{y}_i = \mathbf{E}_i \mathbf{z}$ . The problem we consider is of the form

$$\begin{aligned} & \text{minimize} && \sum_{i=1}^K f_i(\mathbf{x}_i, \mathbf{y}_i) \\ & \text{subject to} && (\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{C}_i, \quad i = 1, \dots, K \\ & && \mathbf{y}_i = \mathbf{E}_i \mathbf{z}, \quad i = 1, \dots, K \\ & && \mathbf{z} \in \mathcal{Z}, \end{aligned} \tag{20}$$

where the variables are  $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1, \dots, K}$  and  $\mathbf{z}$  and  $\mathcal{Z}$  is a *publicly* known convex set that enforces other potential constraints on public variables  $\{\mathbf{y}_i\}_{i=1, \dots, K}$ . We refer  $\mathbf{z}$  as *net variables*.

Primal decomposition techniques allows us to solve the global problem (20) by coordinating  $K$  subproblems, namely one for each entity. Specifically, each entity  $i$  can solve his subproblem, which consists of his own *private* inputs/outputs (e.g.,  $f_i, \mathcal{C}_i, \mathbf{x}_i$ ) together with the public variables  $\mathbf{y}_i$ . The coordination performed via some message exchanges among neighbors. It is interesting to note that, though these messages are dependent on problem input, they are often intrinsically privacy preserving. In other words, the exchanged messages can be such that the input  $\{f_i, \mathcal{C}_i\}$  and output  $\{\mathbf{x}_i\}$  of entity  $i$  is not revealed to other entities  $j \neq i$ . To see this, let us first focus to the derivation of the solution method, where we give a concise discussion.

### Algorithm Development

If we fix the net variables  $\mathbf{z}$  (thus, public variables  $\mathbf{y}_i = \mathbf{E}_i \mathbf{z}$  is fixed), then problem (20) becomes separable into  $K$  subproblems and subproblem for entity  $i$  is given by

$$\begin{aligned} & \text{minimize} && f_i(\mathbf{x}_i, \mathbf{y}_i) \\ & \text{subject to} && (\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{C}_i, \end{aligned} \tag{21}$$

where the variable is  $\mathbf{x}_i$ . We denote by  $\phi(\mathbf{y}_i)$  the optimal value of problem (21). Now note that the original problem (20) is equivalent to the following, which we refer to as the *master problem*

$$\begin{aligned} & \text{minimize} && \phi(\mathbf{z}) = \sum_{i=1}^K \phi_i(\mathbf{E}_i \mathbf{z}) \\ & \text{subject to} && \mathbf{z} \in \mathcal{Z}, \end{aligned} \tag{22}$$

where the variable is  $\mathbf{z}$ . The coordination among the entities resides in the method that is used to solve the master problem (22). Projected subgradient method is a well known algorithm for such



a solution, see [1], [49] for details. In particular, when applying projected subgradient method to solve problem (22), it is required to compute a subgradient <sup>2</sup> of  $\phi$  at a specified  $\mathbf{z}$ . One such specific subgradient is

$$\mathbf{h} = \sum_{i=1}^K \mathbf{E}_i^T \mathbf{h}_i(\mathbf{E}_i \mathbf{z}), \quad (23)$$

where  $\mathbf{h}_i(\mathbf{y}_i) \in \mathbb{R}^{p_i}$  is a subgradient of  $\phi_i$  at  $\mathbf{y}_i$ . This subgradient is obtained for free after solving the subproblem (21). Note that  $\mathbf{h}_i$  is the message, entity  $i$  should send to his neighboring entities. The main algorithm is as follows:

---

*Algorithm 1:* Primal Decomposition

---

Given the initial net variables  $\mathbf{z}$  and iteration index  $k = 1$ .

**repeat**

1. Entity  $i$  sets  $\mathbf{y}_i := \mathbf{E}_i \mathbf{z}$ , solves subproblem (21), and returns subgradient  $\mathbf{g}_i$ ,  $i = 1 \dots, K$ .
  2. Update the subgradient from (23), i.e.,  $\mathbf{h} := \sum_{i=1}^K \mathbf{E}_i^T \mathbf{h}_i$ .
  3. Update the net variables as,  $\mathbf{z} := \Pi_{\mathcal{Z}}(\mathbf{z} - \alpha_k \mathbf{h})$ , where  $\Pi_{\mathcal{Z}}(\cdot)$  denotes the projection onto  $\mathcal{Z}$ . Set  $k := k+1$ .
- 

In this algorithm,  $\alpha_k$  is an appropriate step size. *Algorithm 1* converges to  $\mathbf{x}^* = (\mathbf{x}_1^*, \dots, \mathbf{x}_K^*)$ ,  $\mathbf{y}^* = (\mathbf{y}_1^*, \dots, \mathbf{y}_K^*)$ ,  $\mathbf{z}^*$ , an optimal solution and  $p^*$ , the optimal value of problem (20), respectively (see [1], [32], [49] for details). It is worth noting that each step above is performed in a parallelized manner. This is clearly visible in the case of step 1. In the case of step 2 and 3, each one of them boils down into a communication between neighboring entities, which share a common net. There is *no* need for any entity  $i$  to communicate with entity  $j$ , if they do not have a net in common. In the sequel, we establish some appealing privacy preserving properties of primal decomposition based solution methods.

### Privacy Properties

There are interesting *intrinsic* privacy preserving properties in primal decomposition based methods. They can address cases which are not handled by transformation based methods (§ III). first, it is possible to yield input privacy for the input  $\{f_i, \mathcal{C}_i\}_{i=1, \dots, K}$ , and output privacy for the

<sup>2</sup>Due to page limitations and to avoid information extraneous to the main focus of the paper, we do not go into details here. However, we refer the reader to [1], [50] for details.

output  $\{\{\mathbf{x}_i^*\}_{i=1,\dots,K}, p^*\}$ . This is because *no* entity  $i$  is required to reveal his problem data  $f_i, \mathcal{C}_i$ , as well as the current solution  $\mathbf{x}_i$  to any other entity  $j \neq i$  (see step 1 of *Algorithm 1*). However, to compute the subgradient  $\mathbf{g}$  of the master problem's objective function, entity  $i$  *should* reveal  $\mathbf{h}_i$  (see step 1-2 of *Algorithm 1*). That is the subgradients  $\{\mathbf{h}_i\}_{i=1,\dots,K}$  are the means of coordination between the entities. It is worth emphasizing that these subgradients do not reveal the private information  $\{f_i, \mathcal{C}_i\}_{i=1,\dots,K}$ , and  $\{\mathbf{x}_i\}_{i=1,\dots,K}$  to each other in very many cases. Moreover, note that entity  $i$  requires coordinating or communicating only with its neighbors, who have a common net, in order to carry out the algorithm. As a result, there are inherent boundaries beyond which the information (e.g., subgradient  $\mathbf{h}_i$ ) is not penetrating through multi-party environments. Let us give now some examples to convey these appealing aspects.

---

*Example 13 (Quadratic minimization, a 2-party environment):* Suppose Alice and Bob wants to solve the following problem:

$$\begin{aligned} & \text{minimize} && \mathbf{x}_1^T \mathbf{B}_1 \mathbf{x}_1 + \mathbf{x}_2^T \mathbf{B}_2 \mathbf{x}_2 + 2\mathbf{z}^T (\mathbf{C}_1 + \mathbf{C}_2) \mathbf{z} \\ & \text{subject to} && \mathbf{A}_1 \mathbf{x}_1 \leq \mathbf{z} \\ & && \mathbf{A}_2 \mathbf{x}_2 \leq \mathbf{z} \\ & && \mathbf{x}_1 \geq \mathbf{0}, \mathbf{x}_2 \geq \mathbf{0}, \mathbf{z} \geq \mathbf{0}, \end{aligned} \tag{24}$$

where the variables are  $\mathbf{z} \in \mathbb{R}^p$  and  $\mathbf{x}_i \in \mathbb{R}^{n_i}$ ,  $i = 1, 2$ . The problem data are  $\mathbf{A}_i \in \mathbb{R}^{p \times n_i}$ , and positive semidefinite matrices  $\mathbf{B}_i \in \mathbb{R}^{n_i \times n_i}$  and  $\mathbf{C}_i \in \mathbb{R}^{p \times p}$ , where  $i = 1, 2$ . Suppose Alice wants input privacy for problem data  $\{\mathbf{A}_1, \mathbf{B}_1, \mathbf{C}_1\}$  and Bob wants input privacy for problem data  $\{\mathbf{A}_2, \mathbf{B}_2, \mathbf{C}_2\}$ . It is worth pointing that the transformation based methods (see § III) are *not applied* here. To apply primal decomposition techniques, we first pose this problem in the form of (20):

$$\begin{aligned} & \text{minimize} && \mathbf{x}_1^T \mathbf{B}_1 \mathbf{x}_1 + \mathbf{x}_2^T \mathbf{B}_2 \mathbf{x}_2 + \mathbf{y}_1^T \mathbf{C}_1 \mathbf{y}_1 + \mathbf{y}_2^T \mathbf{C}_2 \mathbf{y}_2 \\ & \text{subject to} && \mathbf{A}_1 \mathbf{x}_1 \leq \mathbf{y}_1 \\ & && \mathbf{A}_2 \mathbf{x}_2 \leq \mathbf{y}_2 \\ & && \mathbf{y}_1 = \mathbf{y}_2 = \mathbf{z} \\ & && \mathbf{x}_1 \geq \mathbf{0}, \mathbf{x}_2 \geq \mathbf{0}, \mathbf{z} \geq \mathbf{0}, \end{aligned} \tag{25}$$

where the private variables are  $\mathbf{x}_i \in \mathbb{R}^{n_i}$ , public variables are  $\mathbf{y}_i \in \mathbb{R}^p$ ,  $i = 1, 2$ , and net variable is  $\mathbf{z} \in \mathbb{R}^p$ . For fixed  $\mathbf{z}$ , problem (25) is decoupled as

$$\begin{aligned} & \text{minimize} && \mathbf{x}_i^T \mathbf{B}_i \mathbf{x}_i + \mathbf{y}_i^T \mathbf{C}_i \mathbf{y}_i \\ & \text{subject to} && \mathbf{A}_i \mathbf{x}_i \leq \mathbf{y}_i \\ & && \mathbf{x}_i \geq \mathbf{0}, \end{aligned} \tag{26}$$

where the variable is  $\mathbf{x}_i \in \mathbb{R}^{n_i}$  (let  $i = 1$  corresponds to Alice and  $i = 2$  corresponds to Bob). By using the usual notation  $\phi(\mathbf{y}_i)$  to denote the optimal value of problem above, we can show that

$$\mathbf{h}_i(\mathbf{y}_i) = 2\mathbf{C}_i \mathbf{y}_i - \boldsymbol{\lambda}_i, \tag{27}$$

where  $\lambda_i$  is an optimal dual variable associated with the first constraint of problem (26). Thus, Alice makes public  $\mathbf{h}_1$  and Bob makes public  $\mathbf{h}_2$  and they communicate to perform *Algorithm 1*. Note that Bob cannot recover the input  $\{\mathbf{A}_1, \mathbf{B}_1, \mathbf{C}_1\}$  of Alice because  $\lambda_1$  is known by Alice only and  $\mathbf{h}_1$  is a transformed variant of  $\mathbf{C}_1$ . Same arguments hold for Alice as well. Thus the solution method yields input privacy for  $\{\mathbf{A}_i, \mathbf{B}_i, \mathbf{C}_i\}_{i=1,2}$ . Moreover, we can easily see that the algorithm yields output privacy for  $\{\mathbf{x}_1^*, \mathbf{x}_2^*\}$ , the solution, and for  $p^*$ , the optimal value of the original problem (24);  $p^*$  is not known to *anyone*.

---

*Example 14 (Linear programming with coupled constraints, a multi-party environment):* Suppose  $K$  entities wants to solve the following problem:

$$\begin{aligned} & \text{minimize} && \sum_{i=1}^K \mathbf{c}_i^T \mathbf{x}_i \\ & \text{subject to} && \sum_{i=1}^K \mathbf{A}_i \mathbf{x}_i \leq \sum_{i=1}^K \mathbf{b}_i \\ & && \mathbf{x}_i \geq \mathbf{0}, \quad i = 1, \dots, K, \end{aligned} \quad (28)$$

where the variables are  $\mathbf{x}_i \in \mathbb{R}^{n_i}$ ,  $i = 1, \dots, K$ . The problem data are  $\mathbf{A}_i \in \mathbb{R}^{p \times n_i}$  and  $\mathbf{c}_i \in \mathbb{R}^{n_i}$ , and  $\mathbf{b}_i \in \mathbb{R}^p$ , where  $i = 1, \dots, K$ . Suppose entity  $i$  wants input privacy for problem data  $\{\mathbf{A}_i, \mathbf{b}_i, \mathbf{c}_i\}$ . We emphasize again that the transformation based method (see § III) are *not applied* here because of  $\{\mathbf{b}_i\}_{i=1, \dots, K}$ . But primal decomposition methods are applied gracefully. To show this, let us first reformulate the problem equivalently as follows:

$$\begin{aligned} & \text{minimize} && \sum_{i=1}^K \mathbf{c}_i^T \mathbf{x}_i \\ & \text{subject to} && \mathbf{A}_i \mathbf{x}_i - \mathbf{b}_i \leq \mathbf{y}_i, \quad \sum_{i=1}^K \mathbf{y}_i = \mathbf{0} \\ & && \mathbf{x}_i \geq \mathbf{0}, \quad i = 1, \dots, K, \end{aligned} \quad (29)$$

where the private variables are  $\mathbf{x}_i \in \mathbb{R}^{n_i}$  and the public variables are  $\mathbf{y}_i \in \mathbb{R}^p$ ,  $i = 1, \dots, K$ . In the problem above, we have not explicitly written out the net variables  $\mathbf{z} = (\mathbf{y}_1, \dots, \mathbf{y}_K) \in \mathbb{R}^{pK}$  and the constraint  $[\mathbf{I}_1 \cdots \mathbf{I}_K] \mathbf{z} = \mathbf{0}$ , where  $\mathbf{I}_i \in \mathbb{R}^{p \times p}$  is the identity matrix. This is to avoid unnecessary notations. For fixed  $\{\mathbf{y}_i\}_{i=1, \dots, K}$ , problem (29) is decoupled as

$$\begin{aligned} & \text{minimize} && \mathbf{c}_i^T \mathbf{x}_i \\ & \text{subject to} && \mathbf{A}_i \mathbf{x}_i - \mathbf{b}_i \leq \mathbf{y}_i, \quad \mathbf{x}_i \geq \mathbf{0}, \end{aligned} \quad (30)$$

where the variable is  $\mathbf{x}_i \in \mathbb{R}^{n_i}$ . With usual notations, we obtain the subgradients as follows:

$$\mathbf{h}_i(\mathbf{y}_i) = -[\mathbf{0}_1, \dots, \mathbf{I}_i, \dots, \mathbf{0}_K]^T \lambda_i, \quad \mathbf{I}_i \in \mathbb{R}^{p \times p}, \quad \mathbf{0}_j \in \mathbb{R}^{p \times p}, \quad j \neq i, \quad (31)$$

where  $\lambda_i \in \mathbb{R}^p$  is an optimal dual variable associated with the constraint  $\mathbf{A}_i \mathbf{x}_i - \mathbf{b}_i \leq \mathbf{y}_i$  of problem (26). Thus, each entity makes public its  $\mathbf{h}_i$  and they communicate to perform *Algorithm 1*. Note that no entity  $j \neq i$  can recover the input of entity  $i$  because  $\lambda_1$  has no information of  $\{\mathbf{A}_i, \mathbf{b}_i, \mathbf{c}_i\}$ . Here, the step 2 and 3 of *Algorithm 1* is simply given by

$$\mathbf{h} := (\mathbf{h}_1, \dots, \mathbf{h}_K), \quad \mathbf{z} := [[\mathbf{I}_1 \cdots \mathbf{I}_K] (\mathbf{z} - \alpha_k \mathbf{h})]^+, \quad (32)$$

where  $\mathbf{I}_i \in \mathbb{R}^{p \times p}$  is the identity matrix and  $[y]^+$  is the projection of  $y \in \mathbb{R}^p$  onto set  $\mathbb{R}_+^p$ , the cone of nonnegative, real  $p$ -vectors. We note that solution method yields input privacy for  $\{\mathbf{A}_i, \mathbf{b}_i, \mathbf{c}_i\}_{i=1, \dots, K}$  as required. We also note that the algorithm yields output privacy for the solution of original problem (48) (i.e.,  $\{\mathbf{x}_i^*\}_{i=1, \dots, K}$ ) and for the optimal value (i.e.,  $p^*$ ), because none can compute it.

### B. Dual Decomposition

Dual decomposition techniques provide another mechanism for solving problems in a parallelized manner. As the name suggests, the machinery behind the dual decomposition is the duality [39, § 5]. They can gracefully handle cases where either the transformation based methods (§ III) or the primal decomposition based methods (§ IV-A) are not applied. In the sequel, we first present concisely the dual decomposition algorithm derivation. Then we discuss the privacy preserving properties of this method followed by some examples.

Here, we use the same problem (20) that we considered in § IV-A. We start by writing the partial Lagrangian of problem (20)

$$L(\mathbf{x}, \mathbf{y}, \mathbf{z}, \boldsymbol{\lambda}) = \sum_{i=1}^K f_i(\mathbf{x}_i, \mathbf{y}_i) + \boldsymbol{\lambda}^T(\mathbf{y} - \mathbf{E}\mathbf{z}) \quad (33)$$

$$= \sum_{i=1}^K (f_i(\mathbf{x}_i, \mathbf{y}_i) + \boldsymbol{\lambda}_i^T \mathbf{y}_i) - \boldsymbol{\lambda}^T \mathbf{E}\mathbf{z} , \quad (34)$$

where  $\boldsymbol{\lambda} = (\boldsymbol{\lambda}_1, \dots, \boldsymbol{\lambda}_K) \in \mathbb{R}^p$  is the Lagrange multiplier associated with  $\mathbf{y} = \mathbf{E}\mathbf{z}$ , and  $\boldsymbol{\lambda}_i$  is the part of  $\boldsymbol{\lambda}$  associated with entity  $i$ . The dual function is obtained by minimizing  $L(\mathbf{x}, \mathbf{y}, \mathbf{z}, \boldsymbol{\lambda})$  over  $(\mathbf{x}, \mathbf{y}, \mathbf{z})$ . For simplifying the presentation, suppose  $\mathcal{Z} = \mathbb{R}^p$ . Thus, the minimization with respect to  $\mathbf{z}$  indicates that  $\mathbf{E}^T \boldsymbol{\lambda} = \mathbf{0}$  (otherwise, the dual function is unbounded below), which acts as a constraint on dual variables  $\boldsymbol{\lambda}$ . Moreover, because the Lagrangian (34) is decoupled, the minimization with respect to  $(\mathbf{x}, \mathbf{y})$  can be carried out independently, i.e., each entity performs minimization with respect to  $(\mathbf{x}_i, \mathbf{y}_i)$ . In particular, entity  $i$  solves the problem

$$\text{minimize } f_i(\mathbf{x}_i, \mathbf{y}_i) + \boldsymbol{\lambda}_i^T \mathbf{y}_i \quad (35)$$

$$\text{subject to } (\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{C}_i ,$$

where the variable is  $(\mathbf{x}_i, \mathbf{y}_i)$ . We denote by  $(\mathbf{x}_i^*, \mathbf{y}_i^*)$  the solution and by  $g_i(\boldsymbol{\lambda}_i)$  the optimal value of problem (35). Then, the dual problem can be formally expressed as

$$\begin{aligned} &\text{maximize } g(\boldsymbol{\lambda}) = \sum_{i=1}^K g_i(\boldsymbol{\lambda}_i) \\ &\text{subject to } \mathbf{E}^T \boldsymbol{\lambda} = \mathbf{0} , \end{aligned} \quad (36)$$

where the variable is  $\boldsymbol{\lambda}$ . Moreover, we can easily show that  $-\mathbf{y}^* = -(\mathbf{y}_1^*, \dots, \mathbf{y}_K^*)$  is a subgradient of  $-g(\boldsymbol{\lambda})$  at  $\boldsymbol{\lambda}$ . We use projected subgradient method to solve the dual problem (36). The algorithm is as follows:

---

**Algorithm 2:** Dual Decomposition

---

Given an initial value of  $\lambda$  such that  $\mathbf{E}^T \lambda = \mathbf{0}$ . Set iteration index  $k = 1$ .

**repeat**

1. Entity  $i$  extracts subvector  $i$  of  $\lambda$ , i.e.,  $\lambda_i$  and solves problem (35). We denote by  $(\mathbf{x}_i, \mathbf{y}_i)$  the solution,  $i = 1 \dots, K$ .
  2. Update the subgradient  $-\mathbf{y} = -(\mathbf{y}_1, \dots, \mathbf{y}_K)$ .
  3. Update the  $\lambda$  as,  $\lambda := \Pi_{\mathcal{S}}(\lambda + \alpha_k \mathbf{y})$ , where  $\Pi_{\mathcal{S}}(\cdot)$  denotes the projection onto  $\mathcal{S} = \{\mathbf{s} \mid \mathbf{E}^T \mathbf{s} = \mathbf{0}\}$ . Set  $k := k+1$ .
- 

Here  $\alpha_k$  is an appropriate step size. As in the case of primal decomposition algorithm, each step of the above algorithm are performed in a parallelized manner as well. Provided some conditions hold true (e.g., strict convexity of  $f_i$ , finiteness of  $f_i$ ), *Algorithm 2* converges to an optimal solution and the optimal value of problem (20). Due to page limitations, we refer the reader to [1], [32], [49] for details. Let us now discuss the privacy preserving properties of distributed optimization methods based on dual decomposition.

### Privacy Properties

Privacy preserving properties of dual decomposition method are very similar to those of primal decomposition based solution methods. For example, the method yields input privacy for the input  $\{f_i, C_i\}_{i=1, \dots, K}$ , and output privacy for the output  $\{\{\mathbf{x}_i^*\}_{i=1, \dots, K}, p^*\}$  (see step 1 of *Algorithm 2*). In particular, *no* entity  $i$  is required to reveal her problem data  $f_i, C_i$ , as well as the current solution  $\mathbf{x}_i$  to any other entity  $j \neq i$ . In order to perform step 2-3 of the algorithm, entity  $i$  *should* however expose the subgradient  $\mathbf{y}_i$ , which is indeed the global variable solution at the current iteration. These subgradients do not expose the private information  $\{f_i, C_i\}_{i=1, \dots, K}$ , and  $\{\mathbf{x}_i\}_{i=1, \dots, K}$  to each other. As for the primal decomposition method, entity  $i$  requires coordinating only with its neighbors, and therefore lesser the information (e.g., subgradient  $\mathbf{y}_i$ ) penetration through the multi-party environment. An example is provided next to highlight these appealing privacy preserving aspects of the dual decomposition method, which is not handled by either the transformation based methods (§ III) or the primal decomposition based methods (§ IV-A).

---

*Example 15 (QP with horizontally partitioned linear constraints, a multi-party environment):* Consider the

following problem that is to be solved by  $K$  entities:

$$\begin{aligned} & \text{minimize} && \sum_{i=1}^K \mathbf{x}^T \mathbf{B}_i \mathbf{x} + \sum_{i=1}^K \mathbf{c}_i^T \mathbf{x} \\ & \text{subject to} && \mathbf{A}_i \mathbf{x} \leq \mathbf{b}_i, \quad i = 1, \dots, K, \end{aligned} \quad (37)$$

where the variable is  $\mathbf{x} \in \mathbb{R}^n$ . The problem data are  $\mathbf{c}_i \in \mathbb{R}^p$ ,  $\mathbf{A}_i \in \mathbb{R}^{q \times p}$ , and *positive definite* matrices  $\mathbf{B}_i \in \mathbb{R}^{p \times p}$ . Suppose entity  $i$  wants input privacy for problem data  $\{\mathbf{A}_i, \mathbf{B}_i, \mathbf{c}_i\}_{i=1, \dots, K}$ . The dual decomposition method gracefully handles the problem. To see this, let us first equivalently reformulate the problem in the form (20), i.e.,

$$\begin{aligned} & \text{minimize} && \sum_{i=1}^K \mathbf{y}_i^T \mathbf{B}_i \mathbf{y}_i + \sum_{i=1}^K \mathbf{c}_i^T \mathbf{y}_i \\ & \text{subject to} && \mathbf{A}_i \mathbf{y}_i \leq \mathbf{b}_i, \quad i = 1, \dots, K \\ & && \mathbf{y}_i = \mathbf{z}, \quad i = 1, \dots, K, \end{aligned} \quad (38)$$

where the variables are  $\mathbf{z}$  and  $\mathbf{y}_i, i = 1, \dots, K$ . Next, each entity  $i$  solves his subproblem (step 1 of *Algorithm 2*):

$$\begin{aligned} & \text{minimize} && \mathbf{y}_i^T \mathbf{B}_i \mathbf{y}_i + \mathbf{c}_i^T \mathbf{y}_i + \boldsymbol{\lambda}_i^T \mathbf{y}_i \\ & \text{subject to} && \mathbf{A}_i \mathbf{y}_i \leq \mathbf{b}_i, \end{aligned} \quad (39)$$

where the variable is  $\mathbf{y}_i$ . Only the solution  $\mathbf{y}_i^*$  is made public by entity  $i$ . Note that  $\mathbf{y}_i^*$  contain no information about  $\{\mathbf{A}_i, \mathbf{B}_i, \mathbf{c}_i\}$ , the private input of the entity. Then all entities communicate to perform step 2-3 of the algorithm. In particular, step 2-3 can be expressed compactly as

$$\boldsymbol{\lambda} := \boldsymbol{\lambda} + \alpha_k (\mathbf{y} - \hat{\mathbf{z}}) \quad \text{or} \quad \boldsymbol{\lambda}_i := \boldsymbol{\lambda}_i + \alpha_k (\mathbf{y}_i - \hat{\mathbf{z}}_i), \quad i = 1, \dots, K, \quad (40)$$

where  $\hat{\mathbf{z}} = (\hat{\mathbf{z}}_1, \dots, \hat{\mathbf{z}}_K) \in \mathbb{R}^{pK}$  and  $\hat{\mathbf{z}}_i = (1/K) \sum_{j=1}^K \mathbf{y}_j \in \mathbb{R}^p$ . Note that this solution method yields input privacy for  $\{\mathbf{A}_i, \mathbf{B}_i, \mathbf{c}_i\}_{i=1, \dots, K}$  as required. We also note that the algorithm yields output privacy for  $p^*$ , the optimal value, because it is not computable during or after the algorithm terminates.

In addition to the classic methods described above (see § IV), there are other related solution methods as well, e.g., the state-of-the art alternating direction method of multipliers [33]. In the sequel, we discuss basic theoretical backgrounds for these methods together with some examples to show their relevance in the context of privacy preserving optimization.

## V. ALTERNATING DIRECTION METHOD OF MULTIPLIERS (ADMM)

As we pointed out in § IV-B, to yield convergence, the dual decomposition method relies on assumptions such as strict convexity of  $f_i$ s and finiteness of  $f_i$ s. ADMM provides an elegant way to exclude such assumptions and thus brings robustness to the dual decomposition method, while still preserving the decomposability of problem (20). Let us next discuss the basic theory behind ADMM. We refer the reader to [33] for details.

The problem formulation is slightly changed, i.e., instead of (general) problem (20), we consider a simple variant that is sufficient to highlight the steps of algorithm development.

$$\begin{aligned} & \text{minimize} && f_1(\mathbf{y}) + f_2(\mathbf{z}) \\ & \text{subject to} && \mathbf{A}\mathbf{y} + \mathbf{B}\mathbf{z} = \mathbf{c} , \end{aligned} \quad (41)$$

where the variables are  $\mathbf{y} \in \mathbb{R}^n$ ,  $\mathbf{z} \in \mathbb{R}^p$  and the problem data is  $\mathbf{A} \in \mathbb{R}^{q \times n}$ ,  $\mathbf{B} \in \mathbb{R}^{q \times p}$ , and  $\mathbf{c} \in \mathbb{R}^q$ . Let us now consider a related problem obtained by adding a quadratic penalty term to the objective function of problem (41), i.e.,

$$\begin{aligned} & \text{minimize} && f_1(\mathbf{y}) + f_2(\mathbf{z}) + (\rho/2) \|\mathbf{A}\mathbf{y} + \mathbf{B}\mathbf{z} - \mathbf{c}\|_2^2 \\ & \text{subject to} && \mathbf{A}\mathbf{y} + \mathbf{B}\mathbf{z} = \mathbf{c} , \end{aligned} \quad (42)$$

where the variables are  $(\mathbf{y}, \mathbf{z})$  and  $\rho > 0$  is called penalty parameter. We can easily show that problem (41) is equivalent to problem (42). Note that the ADMM is associated with the equivalent problem above. By following a similar approach as in § IV-B, we obtain the partial Lagrangian of problem (42) as

$$L(\mathbf{y}, \mathbf{z}, \boldsymbol{\lambda}) = f_1(\mathbf{y}) + f_2(\mathbf{z}) - \boldsymbol{\lambda}^T(\mathbf{A}\mathbf{y} + \mathbf{B}\mathbf{z} - \mathbf{c}) + (\rho/2) \|\mathbf{A}\mathbf{y} + \mathbf{B}\mathbf{z} - \mathbf{c}\|_2^2 , \quad (43)$$

where  $\boldsymbol{\lambda} \in \mathbb{R}^q$  is the Lagrange multiplier associated with  $\mathbf{A}\mathbf{y} + \mathbf{B}\mathbf{z} = \mathbf{c}$ . The minimization of the Lagrangian is performed in *two* steps, as opposed to *one* step in dual approach. In particular, we have 1)  $\mathbf{y}$ -minimization, 2)  $\mathbf{z}$ -minimization. Next, the dual variable update is performed. The algorithm continues in an iterative manner as follows.

---

*Algorithm 3:* ADMM (unscaled)

---

Given an initial variables  $\boldsymbol{\lambda}^{(0)}$ ,  $\mathbf{z}^{(0)}$  and set  $k = 0$ .

**repeat**

1.  $\mathbf{y}^{(k+1)} := \arg \min_{\mathbf{y}} f_1(\mathbf{y}) - \boldsymbol{\lambda}^T(\mathbf{A}\mathbf{y} + \mathbf{B}\mathbf{z}^{(k)} - \mathbf{c}) + (\rho/2) \|\mathbf{A}\mathbf{y} + \mathbf{B}\mathbf{z}^{(k)} - \mathbf{c}\|_2^2$
  2.  $\mathbf{z}^{(k+1)} := \arg \min_{\mathbf{z}} f_2(\mathbf{z}) - \boldsymbol{\lambda}^T(\mathbf{A}\mathbf{y}^{(k+1)} + \mathbf{B}\mathbf{z} - \mathbf{c}) + (\rho/2) \|\mathbf{A}\mathbf{y}^{(k+1)} + \mathbf{B}\mathbf{z} - \mathbf{c}\|_2^2$
  3.  $\boldsymbol{\lambda}^{(k+1)} := \boldsymbol{\lambda}^{(k)} + \rho(\mathbf{A}\mathbf{y}^{(k+1)} + \mathbf{B}\mathbf{z}^{(k+1)} - \mathbf{c})$  .  $k := k + 1$  .
- 

The convergence of the algorithm above is guaranteed with very little assumptions (e.g.,  $f_1, f_2$  are convex, closed, proper), see [33, § 3.2] for details. We skip those due to page limitations. By defining the residual  $\mathbf{r} = \mathbf{A}\mathbf{y} + \mathbf{B}\mathbf{z} - \mathbf{c}$ , a slightly different form of ADMM algorithm can also be obtained [33, § 3.1.1], which is given by

---

**Algorithm 4:** ADMM (scaled)

---

Given an initial variables  $\boldsymbol{\mu}^{(0)}$ ,  $\mathbf{z}^{(0)}$  and set  $k = 0$ .

**repeat**

1.  $\mathbf{y}^{(k+1)} := \arg \min_{\mathbf{y}} f_1(\mathbf{y}) + (\rho/2) \|\mathbf{A}\mathbf{y} + \mathbf{B}\mathbf{z}^{(k)} - \mathbf{c} + \boldsymbol{\mu}^{(k)}\|_2^2$
  2.  $\mathbf{z}^{(k+1)} := \arg \min_{\mathbf{z}} f_2(\mathbf{z}) + (\rho/2) \|\mathbf{A}\mathbf{y}^{(k+1)} + \mathbf{B}\mathbf{z} - \mathbf{c} + \boldsymbol{\mu}^{(k)}\|_2^2$
  3.  $\boldsymbol{\mu}^{(k+1)} := \boldsymbol{\mu}^{(k)} + \mathbf{A}\mathbf{y}^{(k+1)} + \mathbf{B}\mathbf{z}^{(k+1)} - \mathbf{c}$  .  $k := k + 1$  .
- 

### Privacy Properties

To give the general flavor of those properties, suppose  $f_1, \mathbf{A}$  in problem (41) are owned by entity 1 and  $f_2, \mathbf{B}$  are owned by entity 2 and  $\mathbf{c}$  is public. In step 1 of *Algorithm 3* (or 4), entity 2 should make public the *product*  $\mathbf{B}\mathbf{z}^{(k)}$ . Because entity 1 does not have access to  $\mathbf{z}^{(k)}$ , he cannot construct entity 2's private data matrix  $\mathbf{B}$ . Similar arguments holds when entity 2 attempts to construct entity 1's private data matrix  $\mathbf{A}$  as well, see step 2 of the algorithm. Particularized to problem (20), all the privacy preserving properties of dual decomposition method are achieved by ADMM as well. For instance, the method yield input privacy for the input  $\{f_i, \mathcal{C}_i\}_{i=1, \dots, K}$ , and output privacy for the output  $\{\{\mathbf{x}_i^*\}_{i=1, \dots, K}, p^*\}$ . Note that the penalty term here would be

$$(\rho/2) \|\mathbf{y} - \mathbf{E}\mathbf{z}\|_2^2 = (\rho/2) \sum_{i=1}^K \|\mathbf{y}_i - \mathbf{E}_i\mathbf{z}\|_2^2, \quad (44)$$

which is separable among the entities. As a result, the step 1 of *Algorithm 3* (or 4) would be separable and allows each entity  $i$  to solve its' subproblems without revealing to any other entity  $j \neq i$  the problem data  $f_i, \mathcal{C}_i$  and the current solution. The coordination in step 2-3 of the algorithm is through the global variables  $\{\mathbf{y}_i\}_{i=1, \dots, K}$ , which do not expose the private information  $\{f_i, \mathcal{C}_i\}_{i=1, \dots, K}$ , and  $\{\mathbf{x}_i\}_{i=1, \dots, K}$  to each other.

It is important to note that there is a number of other interesting problems where transformation (§ III), primal (§ IV-A), and dual (§ IV-B) based methods do not applies, but ADMM method gracefully applied to solve those in a privacy preserving manner (e.g., problems with functions that takes on  $\infty$ ). In the sequel, we provide some example to give these flavors to the reader.



---

*Example 16 (Intersection of two closed nonempty convex sets):* Suppose Alice owns a private set  $\mathcal{A} \subseteq \mathbb{R}^n$  and Bob owns a private set  $\mathcal{B} \subseteq \mathbb{R}^n$ . They want to find a point  $\mathbf{x}$  such that  $\mathbf{x} \in \mathcal{A} \cap \mathcal{B}$ , without revealing the private sets. We denote by  $f_1$  the indicator function of set  $\mathcal{A}$ , and by  $f_2$  the indicator function of set  $\mathcal{B}$ . Thus, the problem can be posed as

$$\text{minimize } f_1(\mathbf{z}) + f_2(\mathbf{z}) , \quad (45)$$

where the variable is  $\mathbf{z} \in \mathbb{R}^n$ . ADMM method can readily be applied solve this problem in a privacy preserving manner. Consider the equivalent problem

$$\begin{aligned} &\text{minimize } f_1(\mathbf{y}) + f_2(\mathbf{z}) \\ &\text{subject to } \mathbf{y} = \mathbf{z} , \end{aligned} \quad (46)$$

where the variables are  $\mathbf{y} \in \mathbb{R}^n$  and  $\mathbf{z} \in \mathbb{R}^n$ . Thus, given  $\mathbf{z}$  and  $\boldsymbol{\mu}$ , step 1 of *Algorithm 4* simply reduces to returning the solution  $\mathbf{y}^*$  of problem

$$\begin{aligned} &\text{minimize } (\rho/2) \|\mathbf{y} - (\mathbf{z} - \boldsymbol{\mu})\|_2^2 \\ &\text{subject to } \mathbf{y} \in \mathcal{A} , \end{aligned} \quad (47)$$

with variable  $\mathbf{y}$ . The solution  $\mathbf{y}^*$  is simply  $\Pi_{\mathcal{A}}(\mathbf{z} - \boldsymbol{\mu})$ , where  $\Pi_{\mathcal{A}}$  denotes the projection on to  $\mathcal{A}$ . Given  $\mathbf{y}$  and  $\boldsymbol{\mu}$ , step 2 of *Algorithm 4* is identically solved by  $\mathbf{z}^* = \Pi_{\mathcal{B}}(\mathbf{y} - \boldsymbol{\mu})$ , where  $\Pi_{\mathcal{B}}$  denotes the projection on to  $\mathcal{B}$ . finally, step 3 reduces to  $\boldsymbol{\mu}^{(k+1)} := \boldsymbol{\mu}^{(k)} + \mathbf{y}^* - \mathbf{z}^*$ . Clearly, the solution method yields input privacy for  $\{\mathcal{A}, \mathcal{B}\}$ .

---

*Example 17 (Linear programming with partitioned data, a multi-party environment):* Suppose  $K$  entities wants to solve the following problem:

$$\begin{aligned} &\text{minimize } \sum_{i=1}^K \mathbf{c}_i^T \mathbf{z} \\ &\text{subject to } \sum_{i=1}^K \mathbf{A}_i \mathbf{z} \leq \sum_{i=1}^K \mathbf{b}_i , \end{aligned} \quad (48)$$

where the variable is  $\mathbf{z} \in \mathbb{R}^n$ . The problem data are  $\mathbf{A}_i \in \mathbb{R}^{p \times n}$  and  $\mathbf{c}_i \in \mathbb{R}^n$ , and  $\mathbf{b}_i \in \mathbb{R}^p$ , where  $i = 1, \dots, K$ . Suppose entity  $i$  wants input privacy for problem data  $\{\mathbf{A}_i, \mathbf{b}_i, \mathbf{c}_i\}$ . Note that transformation (§ III), primal decomposition (§ IV-A), and dual decomposition (§ IV-B) based methods do not applies here. However, as we will explain, ADMM is readily applied. To show this, let us first reformulate problem (48) as follows:

$$\begin{aligned} &\text{minimize } \sum_{i=1}^K \mathbf{c}_i^T \mathbf{y}_i \\ &\text{subject to } \mathbf{A}_i \mathbf{y}_i - \mathbf{b}_i = \mathbf{z}_i , \quad i = 1, \dots, K \\ &\quad \mathbf{y}_i = \mathbf{z}_{K+1} , \quad i = 1, \dots, K \\ &\quad \sum_{i=1}^K \mathbf{z}_i \leq \mathbf{0} , \end{aligned} \quad (49)$$

where the variables are  $\{\mathbf{y}_i \in \mathbb{R}^n\}_{i=1, \dots, K}$ ,  $\{\mathbf{z}_i \in \mathbb{R}^p\}_{i=1, \dots, K}$ , and  $\mathbf{z}_{K+1} \in \mathbb{R}^n$ . We denote by  $\boldsymbol{\lambda}_i \in \mathbb{R}^p$  and  $\boldsymbol{\mu}_i \in \mathbb{R}^n$  the (scaled) Lagrange multipliers associated with  $\mathbf{A}_i \mathbf{y}_i - \mathbf{b}_i$  and  $\mathbf{y}_i - \mathbf{z}_{K+1}$ , respectively. Thus, given  $\{\mathbf{z}_i\}_{i=1, \dots, K+1}$ ,  $\{\boldsymbol{\lambda}_i\}_{i=1, \dots, K}$ , and  $\{\boldsymbol{\mu}_i\}_{i=1, \dots, K}$ , step 1 of *Algorithm 4* returns  $\mathbf{y}^* = (\mathbf{y}_1^*, \dots, \mathbf{y}_K^*)$ , where  $\mathbf{y}_i^*$  is the solution of problem

$$\text{minimize } \mathbf{c}_i^T \mathbf{y}_i + (\rho/2) \|\mathbf{A}_i \mathbf{y}_i - \mathbf{b}_i - \mathbf{z}_i + \boldsymbol{\lambda}_i\|_2^2 + (\rho/2) \|\mathbf{y}_i - \mathbf{z}_{K+1} + \boldsymbol{\mu}_i\|_2^2 , \quad (50)$$

with the variable  $\mathbf{y}_i$ . Given  $\mathbf{y}$ ,  $\boldsymbol{\lambda}$ , and  $\boldsymbol{\mu}$ , step 2 of *Algorithm 4* returns  $(\mathbf{z}_1^*, \dots, \mathbf{z}_{K+1}^*)$ , the solution of

$$\begin{aligned} & \text{minimize} \quad (\rho/2) \sum_{i=1}^K \|\mathbf{z}_i - (\mathbf{A}_i \mathbf{y}_i - \mathbf{b}_i + \boldsymbol{\lambda}_i)\|_2^2 + (\rho/2) \sum_{i=1}^K \|\mathbf{z}_{K+1} - (\mathbf{y}_i + \boldsymbol{\mu}_i)\|_2^2 \\ & \text{subject to} \quad \sum_{i=1}^K \mathbf{z}_i \leq \mathbf{0}, \end{aligned} \quad (51)$$

with the variable  $(\mathbf{z}_1, \dots, \mathbf{z}_{K+1})$ . Because we want to see the relation of data decomposition in each problem (50), as well as (51), here we do not attempt to find any closed form solutions to those problems. finally, step 3 of *Algorithm 4*, i.e., the (scaled) dual variable update is given by  $\boldsymbol{\lambda}_i^{(k+1)} = \boldsymbol{\lambda}_i^{(k)} + (\mathbf{A}_i \mathbf{y}_i^* - \mathbf{b}_i - \mathbf{z}_i^*)$  and  $\boldsymbol{\mu}_i^{(k+1)} = \boldsymbol{\mu}_i^{(k)} + (\mathbf{y}_i - \mathbf{z}_3)$ ,  $i = 1 \dots, K$ .

Note that problem (50) is solved by entity  $i$ . Therefore, the problem data  $\{\mathbf{A}_i, \mathbf{b}_i, \mathbf{c}_i\}$  of entity  $i$  is not revealed to anyone during the step 1. Moreover, in step 3, entity  $i$  should reveal  $\mathbf{m}_i = \mathbf{A}_i \mathbf{y}_i - \mathbf{b}_i + \boldsymbol{\lambda}_i$  to all the other entities. However, the input  $\{\mathbf{A}_i, \mathbf{b}_i, \mathbf{c}_i\}$  is hidden because no other entity knows changing (in algorithm iterations)  $\boldsymbol{\lambda}$  and as a result, no one can estimate  $\mathbf{A}_i$  and  $\mathbf{b}_i$ . Therefore input privacy is obtained. Moreover, we see that the solution method yields output privacy for the optimal value of the original problem (48).

## VI. DISCUSSION

In this section we compare and summarize the main characteristics of privacy preserving methods for distributed optimization, including cryptographic techniques, transformation based methods, decomposition based methods, and ADMM. The main focus of this comparison is to identify the advantages and the disadvantages of each solution method, thus defining some high level guidelines for possible application in terms of performance indicators such as efficiency, scalability, etc.

Table II summarizes the results of our comparison. We emphasize that this table is to a give *high level* comparison on privacy preserving approaches and is not meant to be comprehensive. Each row of the table is by its own important and can be discussed extensively. However, in the sequel, we highlight key related issues. In the case of first four indicators, we use an integer-scale of 1--5, where, 1 stands for “*low*” and 5 stands for “*high*” and 2--4 stand for in between cases.

Here, the term “efficient” can mean the effective implementation of the protocols in general. Particularized to optimization based methods, efficiency further can mean the fast convergence of the associated iterative algorithms. For many non-trivial problems, cryptographic treatments require very large circuit representations, resulting explosions in associated computation due to integers involving *thousands* of bits [5, § 2.3.2] [29]. However, many optimization methods use

TABLE II  
PRIVACY PRESERVING TECHNIQUES COMPARISON

Performance Indicators	Cryptography	Transformation	Decomposition	ADMM
1. Numerical efficiency	1	5	2-3	4
2. Protocol complexity	5	1	1	1
3. Scalability	1	2-4	5	5
4. Privacy/security level	5	?	?	?
5. Protocol-solver dependence	yes	no	no	no
7. Linear programming restriction	yes	no	no	no
8. Prob. structure and constraint dependence	no	yes	yes	yes
9. Data partitioning dependence	no	yes	yes	yes
10. finite field restriction	no	yes	yes	yes
11. Real field restriction	yes	no	no	no

the state-of-the interior-point solvers [39, § 11], which can be implemented by using floating-point arithmetic and thus they are very effective compared to cryptographic methods. These properties are directly reflected in the case of protocol complexity as well. In terms of scalability, decomposition methods, and ADMM are preferred compared to others. This is because the mechanism associated allows them to coordinate their actions using a *thin* protocol irrespective of the size of the problem. The poor scalability properties of cryptographic methods is mainly due to their low efficiency and high complexity, especially for relatively large problems. Transformation methods also have scalability issues because a transformed variant of the *original problem* has to be solved by each party involved. As a result, the computations are restricted by the size of the problem. There are guaranteed privacy properties in cryptographic methods [6], [34]--[38]. However, in the case of optimization based approaches, there are little investigations for quantifying the privacy. This is still a largely a question mark and an open issue.

In the case of linear programming, cryptographic protocols are often collaboratively executed in each iteration step of the simplex algorithm [5], [44]. Thus, they are inherently restricted to linear programs. More recently the first cryptographic method for handling interior point method have been proposed in [5]. Still the high computational complexity for cryptographically secured iterations in both the simplex and interior point methods is unavoidable. However, the underlying machinery for interior point methods for optimization based approaches are built on linear algebra, which is very efficiently implemented compared to cryptographic treatments. A

basic characteristic of the cryptographic-based methods is that they are carried out over a finite field and thus is constrained to integer domains. In contrast, optimization based approaches are well suited for real vector spaces. finally, note that the cryptographic methods are not much sensitive to the structural changes of the problem, though the protocol is required to be updated. However, the decomposition, and ADMM, methods are highly sensitive to the way the data are partitioned among different parties.

All the distributed privacy preserving methods presented (decomposition, ADMM) in the paper solve the problem in an iterative manner (in the original variable domain) compared to cryptographic and transformation based method. Each iteration requires the entities to share some information about its current state. This collecting of data from several iterations might allow an adversary to estimate private data. This situation can appear in any of the distributed methods. We strongly believe that quantifying the security level in these situations as well as in different optimization approaches is still an open question, requiring a careful discussion of the algorithmic properties as well as the problem assumptions.

## VII. CONCLUSION

In this paper we presented a framework for a systematic study and classification of distributed optimization solvers that are able to preserve the privacy among parties. A general classification of the privacy preserving methods was presented. We summarized the existing privacy preserving techniques by a more general form and we discussed in detail general decomposition structures in optimization and their privacy preserving properties.

Several challenging issues still arise in this field. first, transformation and decomposition methods provide security guarantees that are not strong, especially when applied in small problems. In these cases the disguise methods that are applied do not have enough inputs (variables and constraints) to securely transform the problem to a new domain. Therefore, more research effort must be placed in establishing new methodologies that could guarantee high security levels even in small scale problems. Second, the efficiency of the transformation and the decomposition methods highly depends on the way data are partitioned among the participants in the optimization. We believe that research is required to define a systematic method to disclose

the best partitioning scenario so to reduce the vulnerability of the method. We believe that future parallel and distributed systems that wish to optimally interact while preserving privacy, can gain a significant benefit from the methods we have surveyed. We further believe that the theory of per-se privacy preserving optimization is at a very early stage, an much needs to be investigated.

## REFERENCES

- [1] D.P. Bertsekas and J.N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*, Athena Scientific, Belmont, Massachusetts, USA, 2nd edition, 1997.
- [2] G.D. Stamoulis and J.N. Tsitsiklis, “Efficient routing schemes for multiple broadcasts in hypercubes,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 4, no. 7, pp. 725--739, July 1993.
- [3] O. Goldreich, *The Foundations of Cryptography*, vol. 2, Cambridge University Press, Cambridge, UK, 2004.
- [4] P. A. Forero, A. Cano, and G. B. Giannakis, “Consensus-based distributed support vector machines,” *Journal of Machine Learning Research*, vol. 11, pp. 1663--1707, May 2010.
- [5] A. Bednarz, *Methods for Two-Party Privacy-Preserving Linear Programming*, Ph.D. thesis, Discipline of Applied Mathematics, School of Mathematical Sciences, The University of Adelaide, 2012.
- [6] W. Du and Z. Zhan, “A practical approach to solve secure multi-party computation problems,” in *Proc. New Security Paradigms Works.*, Virginia beach, Virginia, USA, Sept. 23--26 2002, pp. 127--5.
- [7] J. Biskup and U. Flegel, “On pseudonymization of audit data for intrusion detection,” in *Workshop on Design Issues in Anonymity and Unobservability*, NY, USA, Nov. 2000, pp. 161--180.
- [8] J. Biskup and U. Flegel, “Transaction-based pseudonyms in audit data for privacy respecting intrusion detection,” in *Third International Symposium on Recent Advances in Intrusion Detection*, Toulouse, France, Oct. 2000, pp. 28--48.
- [9] W. Diffie and M. E. Hellman, “New directions in cryptography,” *IEEE Transactions on Information Theory*, vol. 22, no. 6, pp. 644--654, Nov. 1976.
- [10] R. Rivest, A. Shamir, and L. Adleman, “A method for obtaining digital signatures and public-key cryptosystems,” *Communications of the ACM*, vol. 21, no. 2, pp. 120--126, Feb. 1978.
- [11] P. Paillier, “Public-key cryptosystems based on composite degree residuosity classes,” in *Advances in Cryptology, EUROCRYPT*, Prague, Czech Republic, May 1999, pp. 223--238.
- [12] B. Schneier, *Applied Cryptography: Protocols, Algorithms, and Source Code in C*, John Wiley and Sons, Inc. 1996.
- [13] M. Rabin, “How to exchange secrets by oblivious transfer,” in *Technical Report Tech. Memo TR-81, Aiken Computation Laboratory*, 1981.
- [14] S. Even, O. Goldreich, and A. Lempel, “A randomized protocol for signing contracts,” *Communications of the ACM*, vol. 28, no. 6, pp. 637--647, June 1985.
- [15] M. Naor and B. Pinkas, “Oblivious transfer and polynomial evaluation,” in *31th ACM Symposium on Theory of Computing*, Atlanta, GA, USA, May 1999, pp. 245--254.
- [16] A. Shamir, “How to share a secret,” *Communication of the ACM*, vol. 2, no. 11, pp. 612--613, Nov. 1979.
- [17] Y. Desmedt, “Some recent research aspects of threshold cryptography,” in *1st International Workshop on Information Security*, Ishikawa, Japan, Sept. 1997, pp. 158--173.
- [18] B. Alomair, A. Clark, J. Cuellar, and R. Poovendran, “Scalable rfid systems: A privacy-preserving protocol with constant-time identification,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 3, no. 8, pp. 1536--1550, Aug. 2012.

- [19] K. Butler, S. Ryu, P. Traynor, and P. McDaniel, "Leveraging identity-based cryptography for node id assignment in structured p2p systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 20, no. 12, pp. 1803--1815, Dec. 2009.
- [20] H. Jinguang, S. Willy, M. Yi, and Y. Jun, "Privacy-preserving decentralized key-policy attribute-based encryption," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 11, pp. 2150--2162, 2012.
- [21] X. Zhang, C. Liu, S. Nepal, S. Pandey, and J. Chen, "A privacy leakage upper-bound constraint based approach for cost-effective privacy preserving of intermediate datasets in cloud," *IEEE Transactions on Parallel and Distributed Systems*, 2012, in press.
- [22] O. Catrina and S. de Hoogh, "Secure multiparty linear programming using fixedpoint arithmetic," in *ESORICS*, Athens, Greece, Sept. 2010, pp. 134--150.
- [23] J. Li and M.J. Atallah, "Secure and private collaborative linear programming," in *International Conference on Collaborative Computing: Networking, Applications and Worksharing*, Atlanta, GA, USA, Nov. 2006, pp. 1--8.
- [24] T. Toft, "Solving linear programs using multiparty computation," *financial Cryptography and Data Security. LNCS*, pp. 90--107, 2009.
- [25] J. Vaidya, "Privacy-preserving linear programming," in *Proc. ACM Symp. on App. Comp.*, Honolulu, Hawaii, USA, Mar. 2009, pp. 2002--2007.
- [26] O. L. Mangasarian, "Privacy-preserving linear programming," *Opt. Letters*, vol. 5, no. 1, pp. 165--172, Feb. 2011.
- [27] O. L. Mangasarian, "Privacy-preserving linear and nonlinear approximation via linear programming," *Opt. Methods and Software*, pp. 1--10, Oct. 2011.
- [28] O. L. Mangasarian, "Privacy-preserving horizontally partitioned linear programs," *Opt. Letters*, vol. 6, no. 3, pp. 431--436, Mar. 2012.
- [29] I. Damgard, "Theory and practice of multiparty computation," in *5th International Conference in Security and Cryptography for Networks*, Maiori, Italy, Sept. 2006, pp. 360--364.
- [30] W. Du, *A Study of Several Specific Secure Two-Party Computation Problems*, Ph.D. thesis, Purdue University, 2001.
- [31] A. Bednars, N. Bean, and M. Roughan, "Hiccups on the road to privacy-preserving linear programming," in *Proc. ACM Works. on Privacy in the Electron. Society*, Chicago, IL, USA, Nov. 2009, pp. 117--120.
- [32] S. Boyd, "Primal and dual decomposition," [Online]. Available: [http://www.stanford.edu/class/ee364b/lectures/decomposition\\_slides.pdf](http://www.stanford.edu/class/ee364b/lectures/decomposition_slides.pdf), 2007.
- [33] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1--122, 2010.
- [34] R. Canetti, "Security and composition of multi-party cryptographic protocols," *J. Crypto.*, vol. 13, no. 1, pp. 143--202, 2000.
- [35] R. Canetti, *Studies in Secure Multiparty Computation and Applications*, Ph.D. thesis, J. Weizmann Institute, Israel, 1995.
- [36] S. Micali and P. Rogaway, "Secure computation," in *Advances in Cryptology - CRYPTO '91*, J. Feigenbaum, Ed. 1991, vol. 576 of *Lecture Notes in Computer Science*, pp. 392--404, Springer-Verlag.
- [37] S. Goldwasser and L. Levin, "Fair computation of general functions in presence of immoral majority," in *Advances in Cryptology - CRYPTO '90*, A. J. Menezes and S. A. Vanstone, Eds. 1990, vol. 537 of *Lecture Notes in Computer Science*, pp. 77--93, Springer-Verlag.
- [38] D. Beaver, "Secure multiparty protocols and zero-knowledge proof systems tolerating a faulty minority," *J. Crypto.*, vol. 4, no. 2, pp. 75--122, 1991.
- [39] S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, Cambridge, UK, 2004.
- [40] O. Goldreich, *The Foundations of Cryptography*, vol. 2, chapter 7, Cambridge University Press, Cambridge, UK, 2004.

- [41] P. Feldman, “A practical scheme for non-interactive verifiable secret sharing,” in *Proc. IEEE Symp. Found. of Comp. Science*, LA, USA, Oct. 1987, pp. 427–438.
- [42] O. Goldreich, S. Micali, and A. Wigderson, “How to play any mental game or a completeness theorem for protocols with honest majority,” in *Proc. ACM Symp. on Theory of Comp.*, NY, USA, May 1987, pp. 218–229.
- [43] J. Dreier and F. Kerschbaum, “Practical privacy-preserving multiparty linear programming based on problem transformation,” in *Proc. IEEE Int. Conf. on Info. Privacy, Secu., Risk and Trust*, Boston, USA, Oct. 2011, pp. 916–924.
- [44] C. Wang, K. Ren, and J. Wang, “Secure and practical outsourcing of linear programming in cloud computing,” in *Proc. IEEE INFOCOM*, Shanghai, China, Apr. 2011, pp. 820–828.
- [45] O. L. Mangasarian and E. W. Wild, “Privacy-preserving classification of horizontally partitioned data via random kernels,” in *Proc. Int. Conf. on Dat. Mining*, Las Vegas, USA, July 2008, pp. 473–479.
- [46] O. L. Mangasarian, E. W. Wild, and G. M. Fung, “Privacy-preserving classification of vertically partitioned data via random kernels,” *ACM Trans. on Knowl. Discov. from Data*, vol. 2, no. 12, Oct. 2008.
- [47] B. Johansson, P. Soldati, and M. Johansson, “Mathematical decomposition techniques for distributed cross-layer optimization of data networks,” *IEEE J. Select. Areas Commun.*, vol. 24, no. 8, pp. 1535–1547, Aug. 2006.
- [48] M. Chiang, S. H. Low, A. R. Calderbank, and J. C. Doyle, “Layering as optimization decomposition: A mathematical theory of network architectures,” *Proceedings of the IEEE*, vol. 95, no. 1, pp. 255–312, Jan. 2007.
- [49] S. Boyd, “Subgradient methods,” [Online]. Available: [http://www.stanford.edu/class/ee364b/lectures/subgrad\\_method\\_slides.pdf](http://www.stanford.edu/class/ee364b/lectures/subgrad_method_slides.pdf), 2007.
- [50] S. Boyd, “Subgradients,” [Online]. Available: [http://www.stanford.edu/class/ee364b/notes/subgradients\\_notes.pdf](http://www.stanford.edu/class/ee364b/notes/subgradients_notes.pdf), 2007.